NEW APPROACHES TO QUALITY TETRAHEDRAL MESH GENERATION

by

Jonathan R. Bronson

A dissertation submitted to the faculty of
The University of Utah
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

in

Computing

School of Computing

The University of Utah

August 2015

The University of Utah Graduate School

STATEMENT OF DISSERTATION APPROVAL

The dissertation of **Jonathan R. Bronson**

has been approved by the following supervisory committee members:

Ross Whitaker	, Chair	**9/12/2014** Date Approved
Mike Kirby	, Member	**9/12/2014** Date Approved
Valerio Pascucci	, Member	**9/12/2014** Date Approved
Adam Bargteil	, Member	**9/12/2014** Date Approved
Joshua Levine	, Member	**9/12/2014** Date Approved

and by **Ross Whitaker**, Chair/Dean of

the Department/College/School of **Computing**

and by David B. Kieda, Dean of The Graduate School.

ABSTRACT

One of the fundamental building blocks of many computational sciences is the construction and use of a discretized, geometric representation of a problem domain, often referred to as a mesh. Such a discretization enables an otherwise complex domain to be represented simply, and computation to be performed over that domain with a finite number of basis elements. As mesh generation techniques have become more sophisticated over the years, focus has largely shifted to quality mesh generation techniques that guarantee or empirically generate numerically well-behaved elements.

In this dissertation, the two complementary meshing subproblems of vertex placement and element creation are analyzed, both separately and together. First, a dynamic particle system achieves adaptivity over domains by inferring feature size through a new information passing algorithm. Second, a new tetrahedral algorithm is constructed that carefully combines lattice-based stenciling and mesh warping to produce guaranteed quality meshes on multimaterial volumetric domains. Finally, the ideas of lattice cleaving and dynamic particle systems are merged into a unified framework for producing guaranteed quality, unstructured and adaptive meshing of multimaterial volumetric domains.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

This work was made possible by the National Institutes of Health/National Center for Research Resources Center for Integrative Biomedical Computing for its support under awards 2P41 RR01125523-12 and 2-P41-RR12553-08, the National Institute of General Medical Sciences of the National Institutes of Health (NIH/NIGMS) for its support under grant number P41GM103545, the US National Science Foundation for its support under awards IIS-1314757 and CCF-073222, and the Department of Energy for its support under grant number NET DE-EE0004449.

CHAPTER 1

INTRODUCTION

Despite many advancements in applied mathematics and computational sciences, one of the most enduring challenges in scientific computing has been the task of decomposing geometric domains into well-formed subregions, or elements. This problem is broadly known as the problem of *mesh generation*, and it comes in many flavors. The work in this dissertation addresses an important class of meshes, those that can be called a simplicial complex. These are meshes composed of triangles in two dimensions, and tetrahedra in three dimensions. In particular, this work focuses on strategies for producing high-quality tetrahedral meshes of domains with piecewise smooth and non-manifold boundaries.

Over the past few decades, the use of computer simulation has grown to be an integral part of practically every scientific field. This has been especially true for mechanical and aeronautical engineering, which have come to rely heavily on the computer-aided design (CAD) pipeline. This pipeline typically begins with an input geometry defined using some form of spline-based representation. While these representations are the preferred choice for modeling piece-wise smooth geometry, computer-aided engineering (CAE) applications including structural analysis and computational fluid dynamics (CFD) often require some form of polyhedral mesh over which to perform computation. Tetrahedral meshes are among the most common. Thus, tetrahedral mesh generation is a key tool in the modern CAD pipeline. Figure 1.1 shows a visualization of an unsteady flow computed across a helicopter using CFD methods. The geometry of the helicopter was designed using conventional CAD modeling, but ultimately a tetrahedral mesh is generated in order to perform the CFD solve.

More recently, the biomedical community has begun to incorporate simulation for biological process modeling and even patient-specific analysis. These models often rely on the computation of solutions to partial differential equations (PDEs) solved using numerical methods. This includes both predictive forward modeling, as well as inverse problems. Thus, converting medical image data to meshes of geometric structures is an increasingly

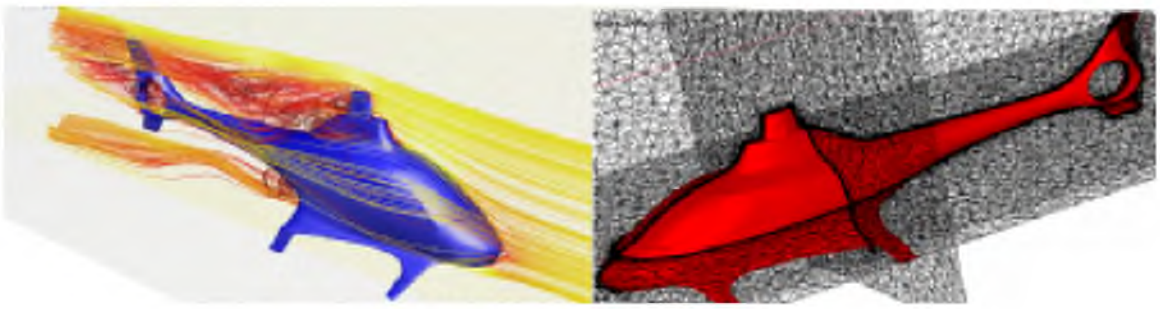


Figure 1.1. A visualization of unsteady flow simulated around a helicopter in forward flight. The geometry of the helicopter was designed using conventional CAD techniques, but a tetrahedral mesh is generated in order to perform the CFD solve [36].

important problem for the medical image analysis community. Figure 1.2 illustrates a mesh of a human brain being used to simulate electromagnetic field propagation in a patient.

The finite element method (FEM) has stood out as one of the most robust methods for solving the necessary partial differential equations (PDEs) on complicated domains. It offers a great deal of flexibility, particularly in regard to discretization, as well as numerical consistency. Among the 3D FEM discretizations, tetrahedral meshes are one of the most favored. They offer a good compromise between generality, simplicity of mesh generation and its implementation, ability to conform to complex geometries, and guarantees both on algorithm termination and output quality.

Due to its impact on numerical simulation, mesh quality has become one of the most important aspects of modern mesh generation. The FEM relies on a weak formulation of a set of PDEs solved as integrals across a set of basis elements that live on an input mesh. Solving these PDEs can be thought of as inverting differential operators in the form of matrices in a linear system. A special matrix known as the *stiffness matrix* is a key part of this linear system, and it is well known that the shapes of the input elements highly impact the numeric stability, or *conditioning*, of this matrix [16, 97]. This conditioning in turn affects both the speed and accuracy of solutions to these systems.

Many measures have been proposed for tetrahedral quality [84], but they all more or less correspond to representing a degree of colinearity and coplanarity among element nodes, or vertices. Mesh size, or element count, also affects speed and accuracy. Finer resolution enables more accurate solutions, but comes at the price of both memory and computational cost. For this reason, mesh adaptivity can also be considered a quality measure. Ideally, a mesh should have high resolution in areas of small features and features of high interest, and low resolution in areas of low curvature or otherwise numerically uninteresting regions.

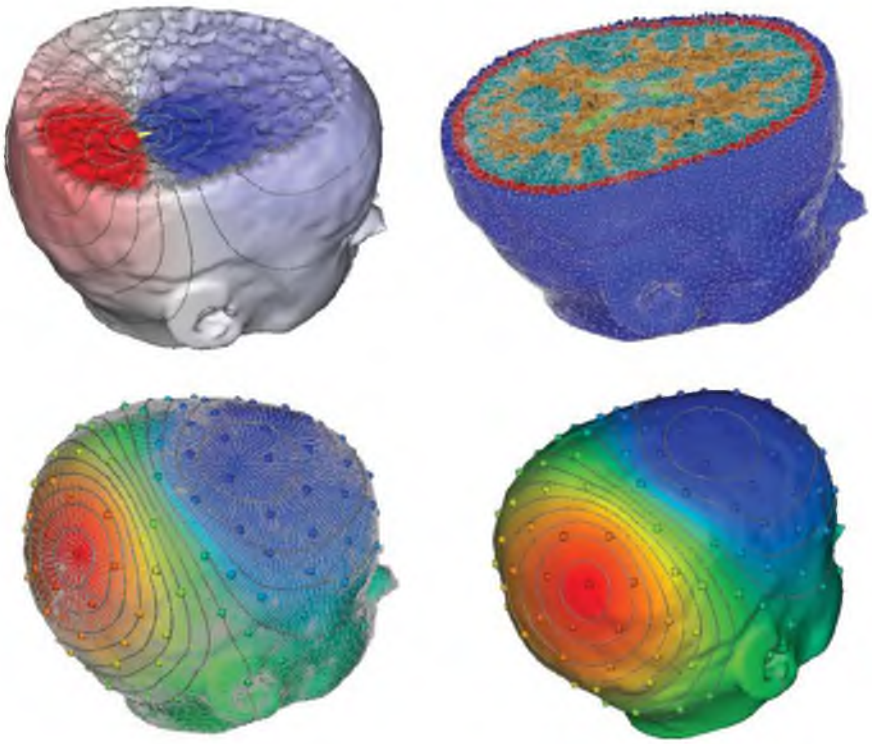


Figure 1.2. Illustration of simulation of electromagnetic field propagation in a patient-specific brain model. The figure shows a finite-element method discretization of Poisson's equation with a patient-specific geometrical model derived from a segmentation of a brain magnetic resonance image (MRI) [70].

In this way, solving PDEs over complicated domains becomes a more tractable problem.

Despite a great deal of important research and many fundamental advances, the general problem of tetrahedral meshing remains unsolved and challenging. In particular, the constraints imposed by adaptive element size, good tetrahedral quality (shape measured by some local metric), and material boundaries are often in conflict, and attempts to satisfy these conditions simultaneously frustrate many conventional approaches, especially those that rely on iterative local updates to the mesh.

Constructing multimaterial tetrahedral meshes that conform to material interface boundaries remains one of the most difficult challenges within tetrahedral mesh generation. In these meshes, each tetrahedron is given a material label, and intermaterial surfaces lie on the boundary triangles between adjacent tetrahedra with differing labels. The tasks of producing an adaptive mesh, high-quality elements, and elements that conform to complex boundaries are often in conflict with one another. To date, there remains no silver bullet solution, and expert users typically mix and match whatever tools are at their disposal.

This dissertation explores several novel approaches for producing high-quality tetrahe-

dral meshes for use in scientific computing and visualization. The particle system and algorithms presented in Chapter 3 achieve topologically robust meshes with high-quality elements on parametric boundary-representation surfaces for CAD models. This formulation introduces the idea of information passing, whereby output elements can infer their local feature size from a combination of locally available information and interaction with adjacent particles. A comparative analysis of angle quality included at the end of Chapter 3 shows that this framework rivals some of the best approaches available.

Variational systems like the one in Chapter 3 can be computationally expensive to solve. Without acceleration they can become prohibitively expensive on large or geometrically complex inputs. Thus, this dissertation also looks into new approaches to combinatoric meshing, which can provide linear time mesh construction through the use of preconstructed lookup tables. In that vein, Chapter 4 presents a new algorithm for generating provably good multimaterial tetrahedral meshes of volumetric datasets.

Building on this, Chapter 5 generalizes the work of Chapter 4 to show that the problem of multimaterial mesh generation can be decomposed into two distinct stages: a stage that generates a background mesh with appropriately sized elements, and a stage that processes this background mesh to conform to material boundaries. This new meshing paradigm is one of the most significant results of this dissertation. Many traditional techniques for creating high-quality elements have difficulties near boundaries. Without a need to process boundaries, these algorithms can be simplified, streamlined, and advanced. Conversely, the generalized cleaving method of Chapter 5 can conform high-quality input meshes to material interfaces with bounded degradation, without requiring the typical highly structured or otherwise restricted input meshes of lattice-based methods. In this way, conforming multimaterial mesh generation no longer necessitates a monolithic algorithm with conflicting constraints, but rather can be seen as a more flexible two-stage pipeline with interchangeable components.

1.1 Contributions

This dissertation provides several new quality meshing techniques for both parametric models as well as multimaterial volumetric domains. To this end, this dissertation provides the following specific contributions:

- *A method for achieving adaptive particle sampling of parameterized domains without an explicit sizing field.* By combining local information with an information passing scheme, a particle system is formulated that optimizes particle positions on model

surfaces while performing most calculations in parameter space. Furthermore, a technique for triangulating these points in parameter space avoids the costly computation that typically takes place in 3D (Chapter 3) [20].

- *A multimaterial tetrahedral meshing algorithm with quality guarantees.* Beginning with a high-quality background lattice, a calculated trade-off between stenciling and mesh warping produces bounded-quality output elements. The technique defines a consistent set of stencil outputs to represent multimaterial topologies. These elements conform to material interfaces and can be calculated efficiently enough for use in time-sensitive applications (Chapter 4) [18].

- *A meshing pipeline for separating element shape and size constraints from surface conforming constraints.* Generalizing the new lattice-based algorithm to arbitrary input meshes breaks the meshing problem into two phases: first producing a background mesh guided by a sizing field, and secondly conforming that mesh to a set of input boundaries (Chapter 5).

- *A new particle system formulation for creating background meshes that conform to a specified sizing field.* This system is ideal for the new decoupled meshing pipeline. Moreover, the absence of surface constraints simplifies implementation and acceleration structures.

- *A set of subdivision strategies for improving the accuracy of visualization for topologically problematic multimaterial datasets.*

1.2 Overview

Chapter 2 provides a short overview of meshing and its context within scientific computing and visualization. That chapter details the mathematical notation and concepts used throughout this dissertation. This includes some fundamental concepts driving modern meshing such as Delaunay triangulation, sizing fields, feature size, as well as metrics for evaluating mesh quality. The main approaches to mesh construction are also summarized to help provide context for the contributions of this dissertation.

Chapter 3 introduces a particle system formulation for generating high-quality tetrahedral meshes of parameterized surface models. After describing the mathematical model of the particles, the notion of an inferred sizing field is introduced, and the various components are discussed in detail. An algorithm for optimizing the formulation is presented with solutions for dealing with corner cases. Next, a method for efficiently triangulating the

particle samples in parameter space is described. The chapter concludes with a comparative analysis of the technique against several competing algorithms. This work is an adapted version of the paper titled "Particle systems for adaptive, isotropic meshing of CAD models", published in Engineering with Computers [17].

Chapter 4 introduces a new guaranteed quality multimaterial tetrahedral meshing algorithm called *lattice cleaving*. After providing related work to put this new technique into context, the concept of indicator functions is introduced. The chapter then describes how these indicator functions interact with a background lattice to define material interface points. Next, a set of tetrahedral stencils are defined that can capture the topology of these material interfaces. A set of geometry and topological rules are then provided for ensuring these output stencils are always high quality. Along with an algorithm for implementing this technique, a theoretical proof of quality bounds is also provided. The chapter concludes with some results of this technique on medical and simulation data. This work is an adapted version of the paper titled "Lattice Cleaving: A multimaterial tetrahedral meshing algorithm with guarantees", published in IEEE's Transactions on Visualization and Computer Graphics [19].

Chapter 5 generalizes the lattice cleaving algorithm to arbitrary unstructured input meshes. This generalization forms the basis of a new mesh generation pipeline. The importance of an accurate sizing field based on local feature size is presented with an approach for generating such a field from input indicator functions. A new particle formulation is formulated that samples a volume based on a sizing field in the absence of material interfaces. The details of generalizing the lattice cleaving algorithm to make use of such an unstructured background mesh are explained, along with a modification to the proof in Chapter 4 that allows the output quality to remain bounded. The chapter concludes with examples of this new meshing pipeline on both structured and unstructured adaptive background meshes. This work is an adapted version of the paper titled "Adaptive and unstructured mesh cleaving", published in the proceedings of the 19th International Meshing Roundtable [21].

Chapter 6 concludes the dissertation with a discussion of the work, including shortcomings and future research. Finally, some early and published results of visualization extensions to the lattice cleaving algorithm are presented. This includes two techniques for resolving and reducing the impact of artifacts that can arise from the method detailed in Chapter 4.

CHAPTER 2

TECHNICAL BACKGROUND

This chapter provides a short introduction to the theoretical underpinnings of mesh generation. Mesh generation is such a large field of research [99] that anything beyond a cursory coverage is outside the scope of this dissertation. Therefore, this chapter is purposefully selective and covers only the topics that are most relevant to the dissertation. First, the mathematical notation used throughout subsequent chapters is defined. Using this notation, the chapter then covers the key ideas of a mesh, triangulation, mesh quality, and feature size. Finally, the major categories of mesh generation algorithms are briefly discussed to give further context to the work in this dissertation.

2.1 Notation

The following mathematical notation is used throughout this dissertation. Logical sets are denoted with upper-case calligraphic letters, (e.g., $\mathbb{R}^2 = \mathbb{R} \times \mathbb{R}$). Matrices are denoted by upper-case, bold letters, while column vectors are denoted by lower-case, bold letters (e.g., $\mathbf{Ax} = \mathbf{b}$). Subscripts indicate an index into a vector or set. For example, $\mathbf{x}_i$ is the ith element of the $\mathbf{x}$ vector. Some additional notation is defined where needed in later chapters.

2.2 Meshes

In 2D, a *tesselation* is a tiling or subdivision of a surface using one or more shapes such that there are no overlaps or gaps between them. In geometric modeling, a connected collection of polygons that tesselate a domain is collectively referred to as a *polygonal mesh*. The term mesh is borrowed from the idea of a physical mesh, an interlaced or woven network of fibers, such as in a net or lattice. This dissertation focuses on simplicial meshes, composed only of simplicial elements. These are triangle meshes in 2D, and tetrahedral meshes in 3D. Formally, a triangle mesh $\mathcal{M}$ embedded in 3D is composed of a set of vertices

$$\mathcal{V} = \left\{ v_1, ..., v_{|\mathcal{V}|} \right\}, \quad v_i \in \mathbb{R}^3 \tag{2.1}$$

connected by a set of edges

$$\mathcal{E} = \left\{ e_1, ..., e_{|\mathcal{E}|} \right\}, \quad e_i \in \mathcal{V} \times \mathcal{V} \tag{2.2}$$

and triangle faces

$$\mathcal{F} = \left\{ f_1, ..., f_{|\mathcal{F}|} \right\}, \quad f_i \in \mathcal{V} \times \mathcal{V} \times \mathcal{V}. \tag{2.3}$$

A tetrahedral mesh is additionally defined by a set of connected tetrahedra

$$\mathcal{T} = \left\{ t_1, ..., t_{|\mathcal{T}|} \right\}, \quad t_i \in \mathcal{V} \times \mathcal{V} \times \mathcal{V} \times \mathcal{V}. \tag{2.4}$$

Meshes capture both the geometry and topology of a surface or set of surfaces that they are meant to represent or approximate. The geometry of the mesh is determined by the location of its vertices. Elements that connect these vertices form piece-wise approximations to the surfaces of the input domain. A mesh is considered *valid* or *well-formed* if it properly tessellates the reference domain and all elements have positive volume. The presence of zero-volume or *degenerate* elements, or overlapping *inverted* elements, render a mesh *invalid*.

We are often interested in knowing whether the surfaces of a mesh are *conforming*. In a conforming mesh, boundary and material interfaces are represented more precisely, with triangle faces aligning with surface tangents and vertices directly sampling boundaries. Figure 2.1 shows a 2D comparison of nonconforming and conforming triangle meshes of a two-material input domain. The same principle applies to tetrahedral meshes in 3D.

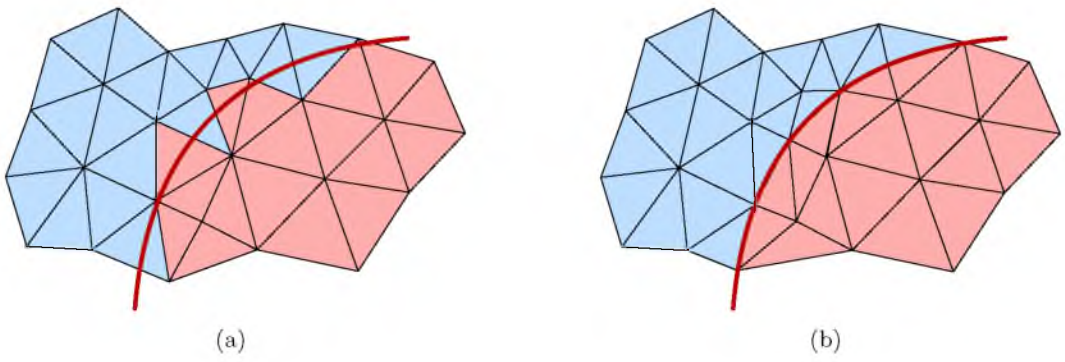


Figure 2.1. A comparison of nonconforming and conforming multimaterial triangle meshes. a) A nonconforming two-material triangle mesh poorly represents the blue-red material interface. b) A conforming two-material triangle mesh accurately approximates the blue-red material interface [103].

The topology of a mesh is related to what is known as the *Euler-Poincaré formula*:

$$v - e + f = 2(1 - g) \tag{2.5}$$

This formula describes the precise relationship between the number of vertices, v, edges, e, and faces, f, in a mesh of *genus*, g. The genus is the number of holes (or *handles*) in the geometry. The right-hand side of this equation is referred to as the *Euler characteristic* and is denoted by the variable χ.

A sphere, for instance, has a genus of zero and an Euler characteristic of two. Similarly, a torus is genus one and has an Euler characteristic of zero. The Euler-Poincaré formula is also generalizable to tetrahedral meshes, by extending the sequence of alternating sums to:

$$\chi = v - e + f - c \tag{2.6}$$

where c is the number of cells (tetrahedra) in the mesh. Computing the Euler characteristic of a mesh is one method for checking that it is a valid tessellation of the input domain.

2.3 Mesh Quality

Mesh quality is a very broad notion referring to a multitude of properties desirable in a mesh. At one level, mesh quality can refer to how well any particular mesh approximates both the topology and geometry of the reference data it is intended to represent. It can also refer to how well the mesh behaves numerically in various settings, such as when used in FEM analysis. In general, it is impossible to determine the quality of a mesh without knowing its intended purpose. Still, there are a limited set of metrics that can be used to determine quality for any particular problem. The quality of a mesh can be decomposed into roughly three categories: element quality, representation accuracy, and structure.

Often when someone refers to the *quality* of a mesh, they are referring solely to its element quality. In triangular and tetrahedral meshes, the quality of an element in isolation is determined strictly by its shape. The shape of an element is often considered in the context of a 2nd-order elliptic PDE operator and those operators have associated metrics. For the purposes of this dissertation, we are only considering isotropic elliptic operators, weighted equally in all directions. Under these operators, *regular* elements are considered ideal, and quality metrics attempt to characterize deviations from these regular triangles and tetrahedra. The case of anisotropic operators and anisotropic meshes is outside the scope of this dissertation.

There are multiple measures that are used to assess element shape quality, but the most commonly used and most easily understood quality measure for simplices is the minimum

and maximum angle: the minimum and maximum planar angles of a triangle, and dihedral angles of a tetrahedron. As the minimum angle approaches 0 degrees or the maximum angle approaches 180 degrees, an element approaches degeneracy (having zero volume). These two extremes cause two types of numerical behavior that are widely known. Small planar angles lead to bad conditioning of the stiffness matrix of the finite element method, whereas large planar angles lead to increased interpolation errors [97, 5]. For other mesh tasks, such as visualization, these issues are usually of less concern, with slight consideration given to interpolation error to avoid distracting lighting artifacts. Aside from minimum and maximum angles, many other metrics are used in the core of meshing algorithms.

The *aspect ratio*, ar, of a triangle or tetrahedron is the ratio of the length of the shortest altitude, a, to the length of the longest edge, l. That is, $r = a/l$. This ratio approaches zero as an element approaches becoming degenerate, and is maximized by regular elements. In later chapters, we will demonstrate a direct relationship between triangle aspect ratio and planar angles as well as between tetrahedral aspect ratio and dihedral angles.

Two other popular and convenient element quality measures are the *radii ratio*, or the ratio of circumcircle and incircle radii ($\rho = R/r$), and the *edge ratio*, or ratio of the length of the longest edge over the length of the shortest edge ($\tau = l/s$). These measures have two properties that make them desirable for a quality measure. First, the measures are nondimensional, and therefore homogeneous across types. Second, they achieve their optimal values at their minimum value of 1 [84].

Yet another two related measures are the *edge-to-circumradius ratio*, and the *edge-to-inradius ratio*. These measures are less intuitive, but are tied to the strategies of some mesh improvement algorithms such as Delaunay refinement, which is reviewed later in this section. The downside to these measures is they do not classify slivers (a type of flat tetrahedron) as poor quality.

Finally, there exists a set of matrix norms that can be used to describe the quality of an element. These norms rely on the edge-matrix of an element. In 2D, this matrix has the following form:

$$T = \begin{bmatrix} x_1 - x_0 & x_2 - x_0 \\ y_1 - y_0 & y_2 - y_0 \end{bmatrix} \tag{2.7}$$

Knupp [60] proposed a series of measures based on the Frobenius norm of the matrix $\mathbf{T}$. Later, Baker [6] generalized these measures to any unitarily invariant matrix norm. While these norms are in some ways more meaningful representations of element quality, they are not often used in the meshing literature.

In addition to the quantitive measures of element quality, qualitative descriptions exist for poorly shaped elements. The two types of skinny triangles are distinguished by the presence of a large obtuse angle (blade) or the lack of it (dagger) (Figure 2.2). Bern et al. [9] and Cheng et al. [24] created taxonomies of badly shaped tetrahedra. The names for these tetrahedra are also given based on the arrangement of vertices and are generally quite descriptive. These classifications can be split into groups: tetrahedra with vertices that are nearly collinear (*spires*, *spears*, *spindles*, *spikes*, and *splinters* (Figure 2.3(a-e))) and tetrahedra with vertices that are nearly coplanar (*wedges*, *spades*, *caps*, *slivers*) (Figure 2.3(f-i)). Some of these elements contribute to error in worse ways than others. The sliver, in particular, is notoriously difficult to eliminate from many otherwise high-quality meshing algorithms.

2.3.1 Feature Size

The quality of a mesh is also based on its topology and geometric fidelity. In terms of topology, the mesh should be homeomorphic to its reference geometry. However, in practice, geometries with high genus may be computational expensive to represent (requiring large meshes to capture small features) and topological simplification (hole closing) is common. In terms of geometric fidelity, triangle and tetrahedral meshes are piecewise linear. For curved surfaces, this means representing surfaces and boundaries will always be an approximation. Therefore, a standard requirement for a quality mesh is that the size of elements be small enough to approximate curvatures adequately, without needlessly increasing the element count of the mesh. This is typically achieved by keeping the size of boundary elements proportional to what is known as the *feature size*, while smoothly grading element size on the interior of domains.

For any point on a surface, the local feature size is defined as the distance to the medial axis of that shape. The medial axis is the set of points that have equal distance to two or more points of the domain [82]. Figure 2.4(a) illustrates a medial axis (sometimes called a

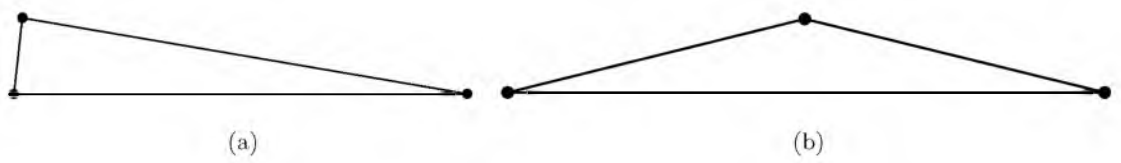


Figure 2.2. Undesirable triangles: (a) A dagger contains one highly acute angle and two near regular angles. (b) A blade contains one large obtuse angle and two highly acute angles.

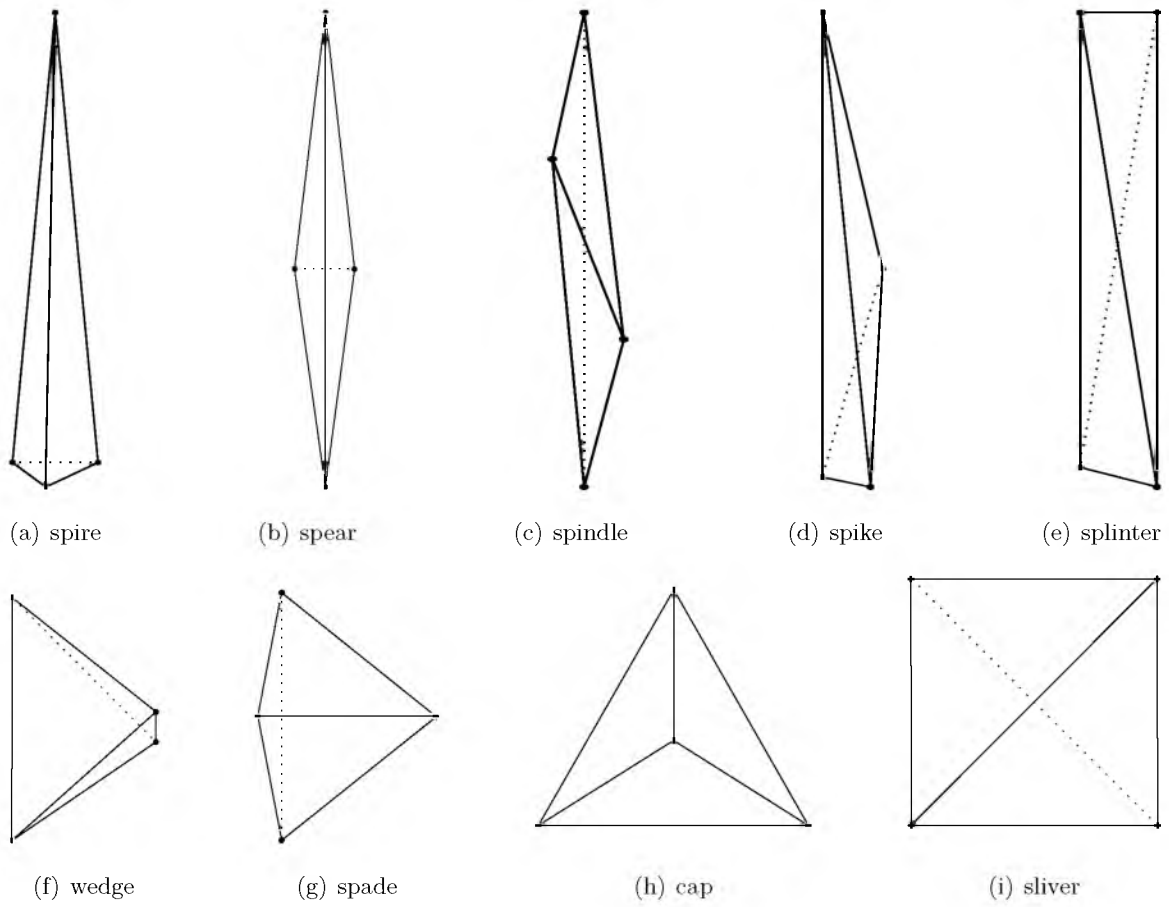


Figure 2.3. Types of undesirable tetrahedra: (a-e) vertices are nearly collinear. (f-i) vertices are nearly coplanar.

skeleton) for a smooth shape. Figure 2.4(b) shows a medial axis for a surface with discontinuities. Note that the medial axis extends to and touches discontinuities on the boundary of the domain. This implies that the local feature size goes to zero at discontinuities. Despite its usefulness for approximating feature size, the robust computation of the medial axis remains a challenging research problem.

The local feature size encapsulates size requirements to both approximate curvature in a consistent way, as well as size needed to accurately capture topology. Thin features have a smaller local feature size, as do regions with a small radius of curvature. Because discontinuities have a feature-size of zero, some form of heuristic is typically used to restrict element size and ensure that the output mesh does not contain an intractable number of elements. Lastly, the structure of a mesh, locally and globally, is also often considered integral to mesh quality. Vertices with a high valence, sometimes called extraordinary points, are undesirable.

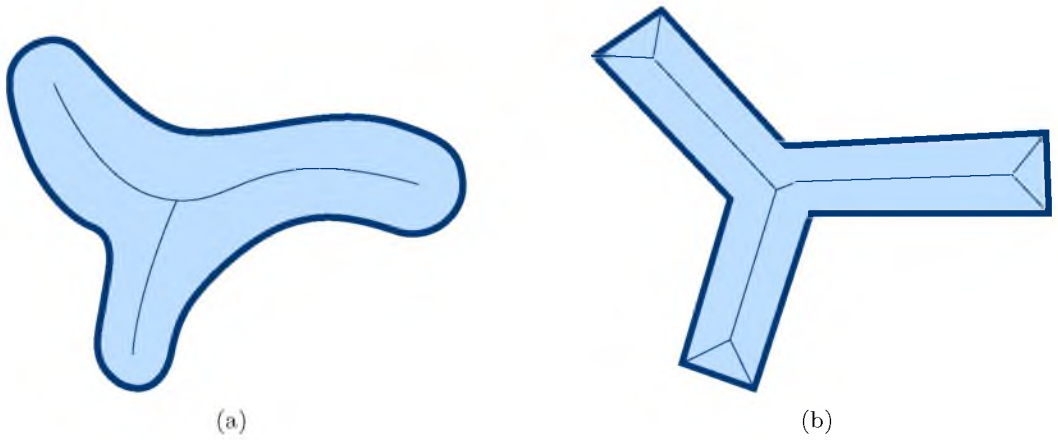


Figure 2.4. Medial axis illustration for a smooth shape (a) and a shape with sharp boundaries (b). The medial axis touches the boundary at discontinuities, causing the local feature size to go to zero. The medial axis is also defined on the exterior of these shapes, but is omitted for simplicity.

2.4 Meshing Approaches

There are four major categories of tetrahedral meshing techniques: Delaunay-based approaches [28, 33, 86, 96], variational optimization methods [1, 47, 63, 75, 108], lattice-based approaches [62, 92, 116], and advancing-front methods [68, 72, 57, 67, 87]. Shewchuk provides a thorough survey of the unstructured meshing approaches [99]. Aside from the lattice-based approaches, the most common strategy for generating surface conforming meshes is to capture surfaces first and worry about volume filling tetrahedra afterward. Some of the techniques go so far as obtaining a fully hierarchical sampling 0 to k-dimensional features [14, 25, 76, 106].

2.4.1 Delaunay

One of the foundational ideas in mesh generation is a special form of triangulation known as a *Delaunay triangulation*. For any set of points in general position, the Delaunay triangulation is a triangulation where no point is inside the circumcircle of any triangle. That is, the circumcircle of every triangle is empty. This special property is called the Delaunay property. The Delaunay triangulation of a set of points is also the dual graph of the voronoi diagram of those points (Figure 2.5). This powerful relationship enabled some of the first provable algorithms for Delaunay mesh generation [13, 27, 26].

In 2D, Delaunay triangulations also have the unique property that they maximize the minimum angle of all triangles in the triangulation. Unfortunately, this highly desirable trait does not extend into 3D. A Delaunay tetrahedralization may contain arbitrarily small angles

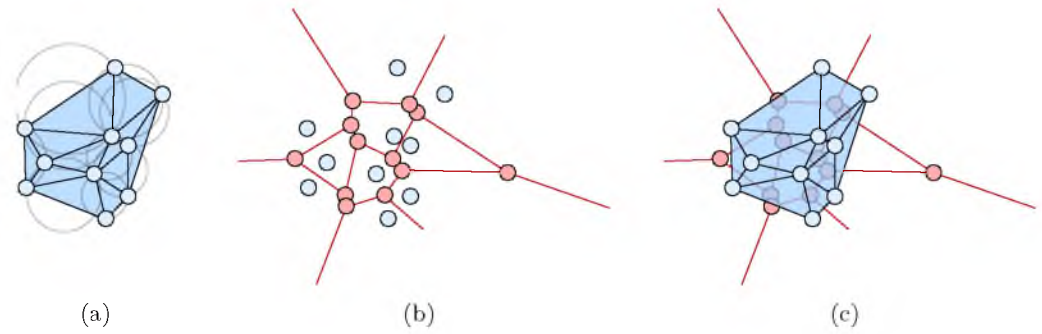


Figure 2.5. Illustration of the relationship between Delaunay triangulations and Voronoi diagrams: a) A Delaunay triangulation b) The dual Voronoi diagram c) The Delaunay triangulation and Voronoi diagram superimposed on one another. Adapted from Okabe et al. [79].

that a non-Delaunay tetrahedralization would not. Another important related concept is the *constrained* Delaunay triangulation. This variation allows an input set of triangles to be fixed, while the rest of the input point set is Delaunay triangulated.

A large amount of research has been focused around the idea of Delaunay refinement [33, 34, 86]. Delaunay refinement methods iteratively insert new points into a Delaunay triangulation until some condition is met. With each insertion, the mesh is modified such that the Delaunay property holds. Figure 2.6 illustrates a single step of an insertion. A candidate triangle of low quality is selected. A vertex is then inserted into the circumscribing ball of that triangle and the mesh retriangulated. This process is repeated until some termination condition has been met, such as a minimum angle rising above a threshold or the largest element decreasing below a threshold.

Point insertion algorithms are locally greedy by nature, and tend to find suboptimal vertex configurations. Protecting features on boundaries is especially difficult under these techniques, and requires simplifying assumptions to work in practice [39, 83]. Delaunay insertion has been adapted for use on segmented medical data [83, 39, 14] and is fairly reliable. Provable guarantees have been given when conforming piecewise linear [96], smooth [13, 27], and piece-wise smooth boundaries [14, 25]. Unfortunately, because they rely on the circumscribing ball property, these methods can still break down in 3D and produce degenerate sliver elements.

2.4.2 Variational Optimization

The poor local configurations that result from Delaunay insertion techniques provided incentive for variational methods to be developed. In these formulations, vertex positions

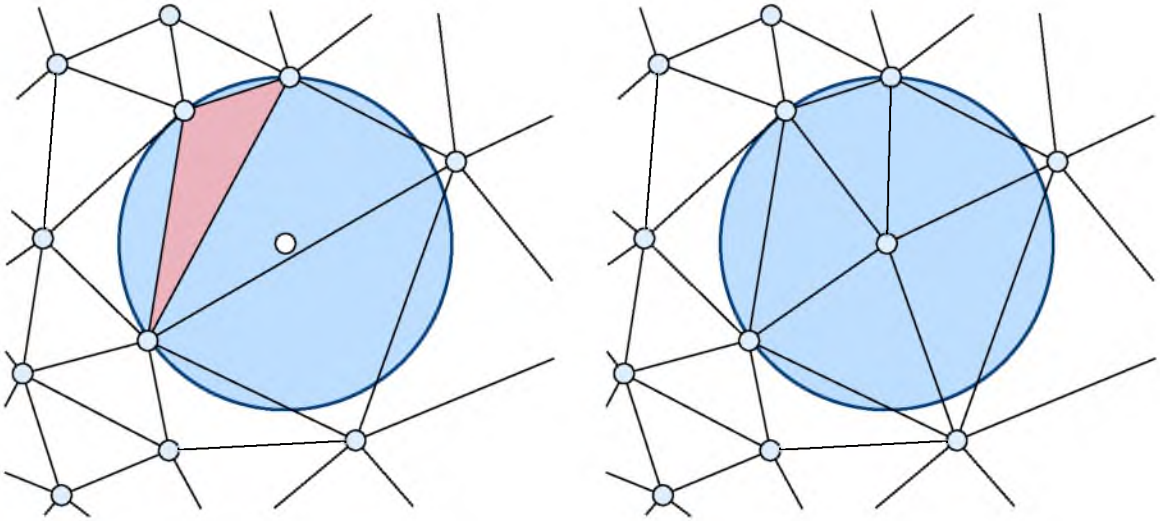


Figure 2.6. An example of Delaunay insertion. A poor-quality candidate triangle is chosen and shown red. Inserting a new vertex within the triangle's circumscribing ball redefines the Delaunay triangulation and destroys the poor-quality triangle. Adapted from Shewchuk [94].

are optimized globally, by optimizing some energy function [76, 106, 113] computed from the vertex positions. While generally more expensive, these formulations tend to produce superior samplings compared to Delaunay refinement. However, they are usually only a mechanism for sampling, and still depend on a triangulation method, which often ends up being Delaunay. Some work has directly blended Delaunay refinement with vertex relaxation to achieve better distributions of points [1, 105, 106, 109, 115]. For instance, centroidal voronoi tessellations are created by iteratively updating generating points until an optimal Delaunay triangulation is achieved [22, 43]. Others have instead used particle systems, which reflect a more physically based approach [76, 113]. Figure 2.7 illustrates four snapshots of a particle system optimizing over a square domain. The initial vertices are seeded randomly, but soon optimize to produce a high-quality triangle mesh. Despite the generally superior mesh quality of variational meshing, these costly methods still cannot guarantee that poor-quality elements do not appear.

2.4.3 Lattice-based

Lattice-based methods take a more combinatoric approach to the problem of mesh construction. Rather than attempting to capture surfaces and construct a volume bounded by these surfaces, lattice-based techniques begin with a volumetric structure of some kind, and warp or stencil it to insert elements and match boundary conditions. Stencils vary in

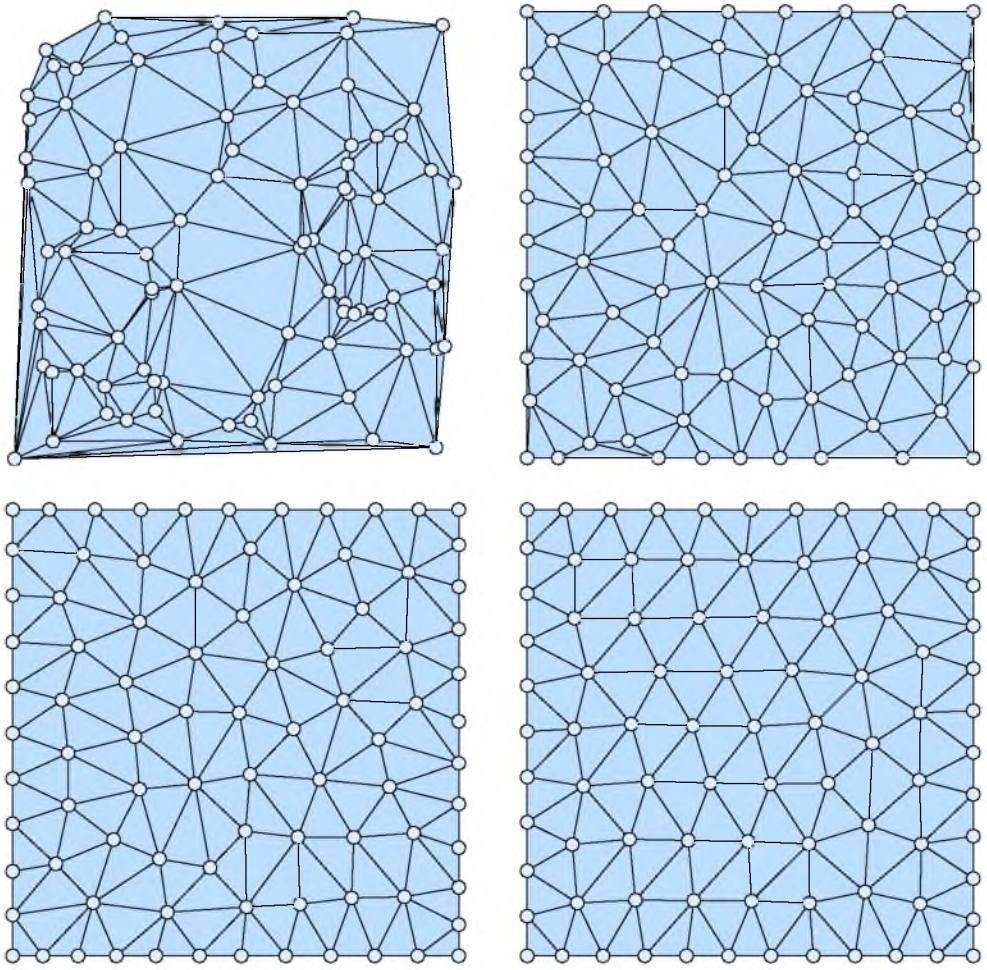


Figure 2.7. A particle system uses repulsive forces to distribute samples over a square domain. Delaunay triangulation is used to construct a mesh from these samples. Rather than keeping the location of vertices as new points are added, variational methods iteratively update the position of vertices until the quality of the mesh is above some threshold.

topology to match the configuration with which they intersect a background lattice cell. This strategy was popularized by the well known marching cubes algorithm [69]. Figure 2.8 shows the 15 basic topologies used in this technique. The success and failure (arbitrarily bad elements) of marching cubes led to further development in lattice-based methods, with a variety of strategies. For instance, Zhang et al. [120] use a pillowing strategy on octree cells, while Liu et al. [65] use a two-step process to mesh a background grid.

Regular structures can often be adapted to provide a natural adaptivity by using spatial subdivision structures, such as octrees [116, 92]. The body-centered cubic (BCC) lattice is one of the best regular structures for tetrahedral meshes, because it is conveniently patterned with identical high-quality tetrahedra. It has been shown that this lattice is particularly

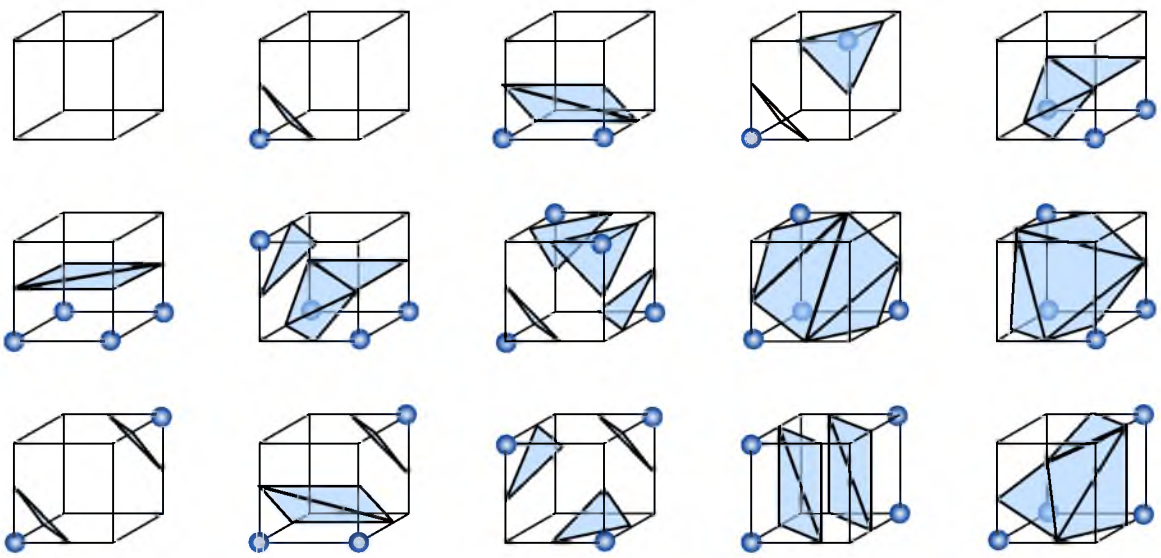


Figure 2.8. The 15 basic topologies of the marching cubes algorithm. Each topology is identified and a lookup table is used to obtain the precomputed tessellations needed for each case. Adapted from Lorensen and Cline [69].

good at approximating trivariate functions [54]. It has been the basis of several high-quality lattice algorithms. The isosurface stuffing algorithm [62] utilizes a careful set of *warping* and *snapping* rules to both stencil and simplify the BCC lattice. The authors used integer arithmetic to place tight numerical bounds on the worst quality outputs. Unfortunately, this technique only works for single smooth boundaries, or isosurfaces. Approaches for achieving multiple surfaces often employ successive subdivision [65, 120, 77]. While these techniques tend to give good results, they offer no formal guarantees on quality. This is one of the primary motivations behind the work in this thesis.

2.4.4 Advancing Front

Rather than decomposing an input domain into a regular structure or trying to solve for a mesh globally, advancing front techniques [68, 72, 57, 67, 87] operate via a greedy strategy. Beginning with an initial set of points or elements, a mesh wave-front propagates over the input domain. To advance the front, new candidate vertices are picked by some ranking function and connected to the existing mesh boundary. When the front is empty, the mesh is completed. Figure 2.9 illustrates an advancing front mesh being generated over a small 2D surface. The selection of new candidate vertices is one of the most important operations of these methods. The choice of location should not only create a new optimally shaped element along the front, but also do it in a way that does not limit the choices of neighbor elements in the front.

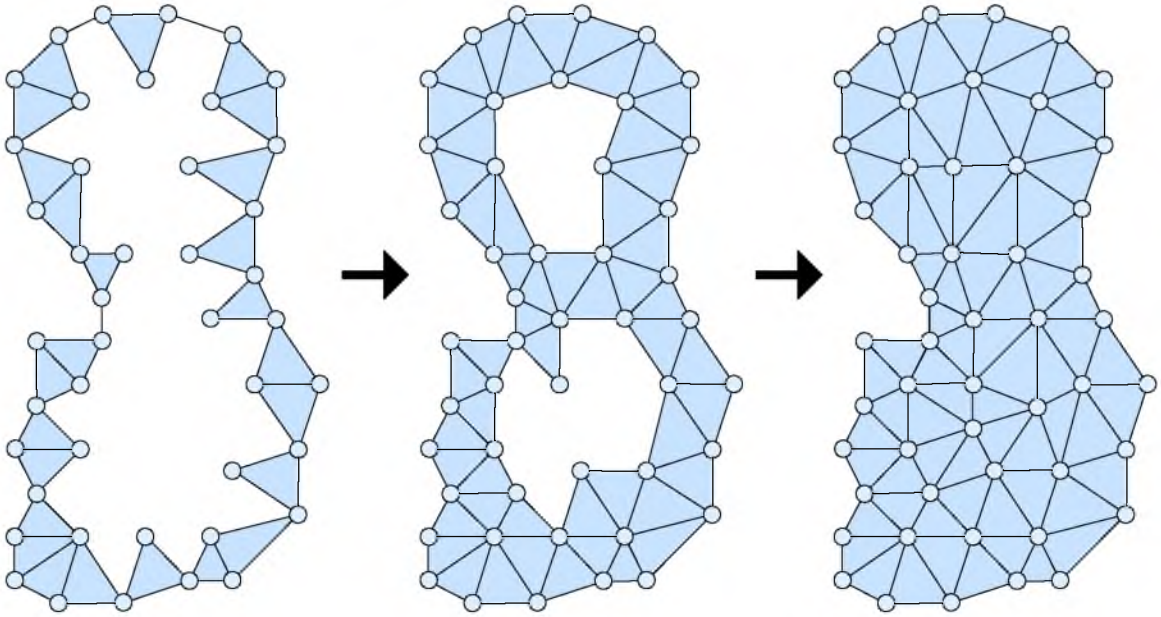


Figure 2.9. An illustration of an advancing front meshing technique. At each iteration, a wave-front of already triangulated elements is the starting point for the next set of adjacent elements. These new elements are added in a greedy fashion, propagating the mesh wave over the input geometry. Adapted from Shewchuk [99].

Advancing front meshes are well suited as input to computational methods that require boundary layers, as the fronts naturally admit aligned layers of elements that can be sized based on the distance to geometry surfaces. These methods also often assume that the solution gradient is largest at the boundary. Some advancing front techniques make use of a guidance field to dictate element size along the wave front [88, 89, 87].

These techniques are typically efficient along the wavefront, as there is no need refine or remesh regions over which the wavefront has already passed. This also makes them especially amenable to streaming and out-of-core implementations. However, problems arise when wavefronts collide. Special rules must be applied to resolve their intersections and inconsistencies. This can lead to characteristic meshes with high-quality elements throughout most of the domain, but with seams of low-quality stitched elements. The more complex an input domain or geometry, the more wave-fronts are likely to interact with one another to create these seams.

CHAPTER 3

ADAPTIVE PARTICLE SYSTEMS

This chapter presents a particle-based approach for generating adaptive triangular surface and tetrahedral volume meshes from boundary representation (B-Rep) CAD models. In such models, input shapes are treated as a collection of smooth, parametric surface patches that can meet nonsmoothly at their boundaries. The approach uses a hierarchical sampling scheme that places particles on features in order of increasing dimensionality. These particles reach a good distribution by minimizing an energy computed in 3D world space, with movements occurring in the parametric space of each surface patch.

One of the contributions of the system described in this chapter is algorithmic control for both uniform and adaptive sampling, *without* requiring a precomputation of global feature size needed by similar particle-based approaches [76]. Rather than using a precomputed measure of feature size, the system automatically adapts to both curvature as well as a notion of topological separation. It also enforces a measure of smoothness on these constraints to construct a sizing field that acts as a proxy to piecewise-smooth feature size. We evaluate the technique with comparisons against other popular triangular meshing techniques for this domain [25, 51].

3.1 Background

While the history of tetrahedral mesh generation began much earlier, a shift in the emphasis of techniques has become popular within the past decade. In particular, variational approaches, i.e., based on energy minimization, have become one of the most important tools for mesh generation. Alliez et al. [1] describe a variational technique for mesh generation that couples Delaunay refinement with a relaxation process for vertex locations. This algorithm and later variants [105, 106, 109, 115] base their energy minimization on a sizing field for particle density coupled with an energy minimization grounded in the notion of a *centroidal Voronoi diagram* [43] and its dual, the *optimal Delaunay triangulation* [22]. Consequently, these meshing algorithms can generate nearly isotropic elements in 2D and

3D, as a byproduct of the centroidal Voronoi condition, as well as leveraging many of the benefits of Delaunay refinement techniques.

However, one deficiency is the need for knowledge of an element sizing field *a priori*. Computing a sizing field is considered expensive. Often, approaches for computing sizing fields are based on the medial axis [41] or quadratures of mesh elements [4], and thus can require $O(n^2)$ computations of dense point clouds to build accurate results. One recent solution of Tournois et al. [106] solves this problem by alternating a variational phase with a refinement phase. After each level of refinement, the sizing function is updated before switching back to variational phase. This interleaving allows the available information to drive the computation of a sizing field instead of necessitating a preprocessed computation. We aim to improve upon this theme by allowing an energy minimization based on particle systems to automatically improve its approximation of the sizing field.

Many techniques simply require an input oracle that evaluates a sizing field over the domain [13, 26, 76, 89, 106]. An early exception is the approach of Dey et al. [40] that uses Delaunay refinement for meshing smooth domains. Using the dual Voronoi diagram and the concept of *poles* [2], this algorithm automatically refines based on a simultaneously computed approximation of the *local feature size* (distance to the medial axis) of the shape whose accuracy increases as mesh density increases.

The local feature size of smooth shapes is a natural choice for guiding the adaptivity of element size; however, most CAD models are inherently nonsmooth. A notion of local feature size for piecewise-smooth shapes has been defined [26] by coupling local feature size for the smooth regions with a topological condition called *gap size* [29]. Computing this measure robustly is a significant challenge. The approach in this chapter aims to automatically infer a global sizing field of equivalent expressivity to [26] while using only locally available information as done by [40]. Such a technique must force a compromise. Here, it is to construct a proxy for feature size that is Lipschitz continuous by coupling curvature adaptivity with a topological separation condition.

A second thrust of recent algorithms is to provide provably algorithms for meshing piecewise-smooth shapes. This general class describes shapes with a topological description in the form of a piecewise-smooth complex of k-cells that are compact subsets of k-manifolds. We use the same definition as Cheng et al. [26]. In summary, surface patches (2-cells) can meet nonsmoothly at curves (1-cells) bounded by points (0-cells). Two k-cells are *adjacent* if one is on the boundary of the other.

Similar to the B-Rep definition, each k-cell has an associated geometric description.

Recent Delaunay-based approaches [26, 85] for meshing this domain have been able to provide topological correctness guarantees using either weighted Delaunay triangulations [26] or bounding the angle deviations between smooth patches [85]. A missing piece to the implementations of these algorithms is the ability to adapt to a sizing field, primarily because there is no consensus on what is the correct sizing field for nonsmooth shapes and how best to compute it. However, they do show that a careful sampling of points on sharp creases can preserve the features of a shape. The approach in this chapter is a natural extension of this work, but instead of requiring an accurate sizing field to guarantee topological correctness, the scheme will build watertight meshes provided a few easily satisfied conditions are met by the particle system (described in Section 3.4).

At the core of the meshing scheme is a paradigm for sampling shapes using particles. The idea of using repulsive point clouds to (re-)sample a mesh was first introduced by Turk in the context of polygonal remeshing [108]. The first full particle system for meshing was later developed by Witkin and Heckbert [112]. Their technique was primarily used as a mechanism to sample and control implicit surfaces, which was notoriously difficult under other schemes at the time. The key idea behind their work was the introduction of a Gaussian energy function to control the interaction between particles. Improvements to their scheme were made by Hart et al. [53]. Yamakawa and Shimada proposed a meshing scheme similar to particles by using packings of ellipsoidal bubbles [113].

Meyer et al. [74, 76, 75] formulated a more robust and stable solution for evolving particle systems. The new energy kernel was a modified cotangent function, with finite support. By falling off to a finite range, the resulting particle systems were more stable and more quickly lead to ideal packings. Additionally, this kernel was nearly scale invariant. Meyer et al. [76] later introduced a hierarchical scheme for particle-based sampling multimaterial surfaces. For such datasets, the boundaries between the different materials can be represented as a piecewise-smooth complex. While without the formal guarantees of [26], they use a similar strategy of hierarchically sampling topologically features in increasing dimension to build consistent, watertight meshes.

3.2 Formulation

In this section, we provide the mathematical formulation behind the proposed particle system. We define the total energy in the system as the sum of each energy E_i calculated with respect to particle p_i. Each particle p_i has a corresponding σ_i value representing the radius of its ball of influence B_i centered at p_i. It is the varying of σ_i that provides adaptivity.

Each energy E_i is the sum of the energies between particle p_i and all neighboring particles p_j. Particles p_i and p_j are considered neighbors if either p_j falls within B_i or if p_i falls within B_j. We use a variation of the modified cotangent for the energy (3.1) between any two particles, E_{ij}. By varying σ_i, the potential function must be scaled to account for this new, lopsided interaction between particles. Thus, we scale both the modified cotangent function and its derivative (3.2) by σ_i.

$$E_{ij} = \sigma_{ij} \cot\left(\frac{|r_{ij}|}{\sigma_{ij}}\frac{\pi}{2}\right) + \frac{|r_{ij}|}{\sigma_{ij}}\frac{\pi}{2} - \frac{\pi}{2} \tag{3.1}$$

$$\frac{dE_{ij}}{r_{ij}} = \frac{\pi}{2}\left[1 - \sin^{-2}\left(\frac{|r_{ij}|}{\sigma_{ij}}\frac{\pi}{2}\right)\right] \tag{3.2}$$

In this form, $|r_{ij}|$ is the distance between particles p_i and p_j and the value σ_{ij} is taken to be the max of σ_i and σ_j. The hexagonal packings that result from this and related particle systems requires the particles to reach a critical density on the surface being sampled. For any surface and any set of σ values, there will always be an ideal number of particles, but calculating this number is not tractable. Like previous systems, we use splitting and deleting to control energy densities. Particles follow the rules:

$$E_i^* = E_i\,(1+\epsilon) \tag{3.3}$$

$$\text{if } E_i^* < 0.35E_i^{\text{ideal}} \qquad \text{Split} \tag{3.4}$$

$$\text{if } E_i^* > 1.75E_i^{\text{ideal}} \qquad \text{Delete} \tag{3.5}$$

where the coefficients 0.35 and 1.75 are used to bias the rate of particle splitting and deletion. In practice, these values provide a good balance between stocastic behavior and stability, helping to prevent the system from getting stuck in local minima.

Using a hexagonal packing as the notion of an ideal distribution, the ideal energy E_i^{ideal} for a particle p_i is six times the energy felt between p_i and p_j at the characteristic distance of approximately 0.58 [74]. Given that a two-ring particle p_j is at distance 1.0, Equation (3.6) describes this relationship. Additionally, we scale this value by σ_i to match the scaling of actual energies.

$$E_i^{\text{ideal}} = \sigma_i 6E(\beta), \quad \text{with } \frac{|r_{ij}|}{\sigma_{ij}} = \beta = \frac{0.5}{\cos(\pi/6)} \approx 0.58 \tag{3.6}$$

Because one cannot predict what an ideal neighborhood will look like in the adaptive case, the ideal energy is less precise than in the constant case. This leads to more frequent splits and deletes for higher local variation, but ultimately provides much better packings than if the original energy was not scaled proportional to σ. An alternative to this approach

would be to use a notion of scaled distance $d' = \frac{d}{\sigma}$, and forego the σ_i scaling. Then, to still achieve the high-quality packings, a different scheme for deletion of poorly configured particles would need to be devised.

To allow the system to adapt to splits and deletes, E_i is biased by a small random number, $0 \leq \epsilon \leq 1$, in Equation (3.3). This makes the discrete energy jumps have less of an impact on the speed at which the system stabilizes, by allowing time for the system to adapt between jumps. Additionally, this can help resolve any regions which are stuck in bad configurations. As the solution to the system converges, this bias can be adjusted to stop splits and deletes all together, ensuring termination.

To find the ideal packing of particles, we use a Newton-Raphson scheme, updating particle information after each movement (Equations (3.7), (3.8), and (3.9)). Each particle maintains its position in both world space (x_i^{xyz}) and parameter space (x_i^{uv}). Particles move with a velocity $\mathbf{v}_i$ generated by interparticle forces between neighbors. Though energies between particles are computed in 3D world space, particles move strictly in parametric space (3.9), avoiding the error-prone projection onto the surface that results from 3D movements. Taking these steps in parameter space only requires a change of coordinates, using the inverse Jacobian, $\mathbf{J}^{-1}$.

$$\mathbf{v}_i^{xyz} = \sum_{j \in \mathcal{N}} dE_{ij} \cdot \frac{r_{ij}}{|r_{ij}|} \tag{3.7}$$

$$\mathbf{v}_i^{uv} = \mathbf{v}_i^{xyz} \cdot \mathbf{J}^{-1} \tag{3.8}$$

$$\mathbf{x}_i^{uv} = \mathbf{x}_i^{uv} + \mathbf{v}_i^{uv} \tag{3.9}$$

As mentioned earlier, we use a hierarchical sampling scheme, which works well for parametric models. First, we place particles on the 0-cells, the intersection of edges on the models. Next, we place particles on the 1-cells and allow them to be optimized. Finally, we place particles on the surface patch interiors and the final optimization proceeds. At each phase, the new optimization uses the fixed positions from the previous phase, ensuring consistency across surface patch boundaries.

3.2.1 Inferred Sizing Field

We recognize that there are several factors that often determine good sizing fields: local curvature, some notion of feature size, and a desired level of adaptivity. Additionally, users may have desires for mesh resolution limits, both minimum and maximum triangle or edge size. Other domain-specific factors also often come into play. In this section, we illustrate the constraints we would like to place on a sizing field. We show that these constraints

can be inferred in a reliable way and used to form a smooth sizing field during energy minimization.

We aim for meshes that provide arbitrary levels of geometric accuracy and adaptivity, using high-quality isotropic elements. In order to provide high-quality elements, particle systems require enough spatial freedom to be able to move to lower energy states. Thus, the distance between nearby k-cells imposes its own sizing constraint on the particles. Thus, in order to determine the sizing field value σ_i at a particular point p_i on a model, we must consider the constraints placed on this location by curvature, topological distance, and desired level of adaptive continuity. We refer to these constraints as σ_κ, σ_τ, and $\sigma_\mathcal{L}$, respectively. The actual sizing field value at a particle location is resolved by finding the σ_i that respects all constraints. This can be expressed compactly as:

$$\sigma_i = \max\{\sigma_{\min}, \min\{\sigma_{\max}, \sigma_\kappa, \sigma_\tau, \sigma_\mathcal{L}\}\} \tag{3.10}$$

3.2.1.1 Curvature

Because the curvature at a point is defined as the inverse of the radius of the osculating circle at that point, a reasonable default sizing field value is the radius of that circle itself (Figure 3.1). Thus, we use $\sigma_\kappa = \frac{1}{\kappa}$, which can be easily computable for parametric surfaces, or queried by middleware packages.

To increase or decrease the field relative to this radius, a scaling factor s_κ is exposed as a user parameter. Given a unit system, this value can be used to provide constraints to respect geometry to arbitrary levels of accuracy. Finally, $\kappa_{\min}$ and $\kappa_{\max}$ values are user parameters used to handle straight edges and arbitrarily high curvature, respectively.

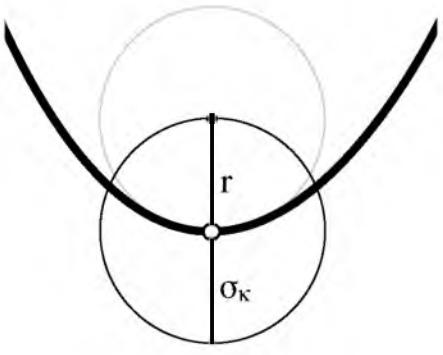


Figure 3.1. Default curvature constraint on sizing field.

These form the total bounds for the sizing field as:

$$\sigma_{\min} = 1/(s_\kappa \kappa_{max}) \tag{3.11}$$

$$\sigma_{\max} = 1/(s_\kappa \kappa_{min}) \tag{3.12}$$

For 2-cells, we use the maximum absolute value principal curvature, because this size will dominate an isotropic sampling. For 1-cells, using the curvature of the edge itself is insufficient. The maximum principal curvature on both intersecting surfaces must also be considered, because the curve may either be a trim or a boundary curve, and there is no way of knowing which curvature will dominate. Last, 0-cells use the maximum curvature of all 1-cells terminating at its point.

Figure 3.2 illustrates the intuitive effect of modifying the curvature scaling parameter s_κ to achieve a better geometric fit. For practical purposes, a good scaling parameter will usually be based on the unit of measurement of the model.

3.2.1.2 Gap Size

If available, using the distance to the model's medial axis would provide a sizing field constraint that generates good samplings in a particle system. However, computing the medial axis on parametric models is a difficult task and still an active area of research. Instead, we use the notion of *gap size*, introduced by Chang and Poon [29] in the context of piecewise linear mesh generation. For a point p on a k-cell c, its gap size is the distance to the nearest nonadjacent (i.e., not on the boundary of c) cell. This measure also preserves topological features inherent to the model's decomposition into parametric patches. Depending on the model and the way it was created, this measure may sometimes

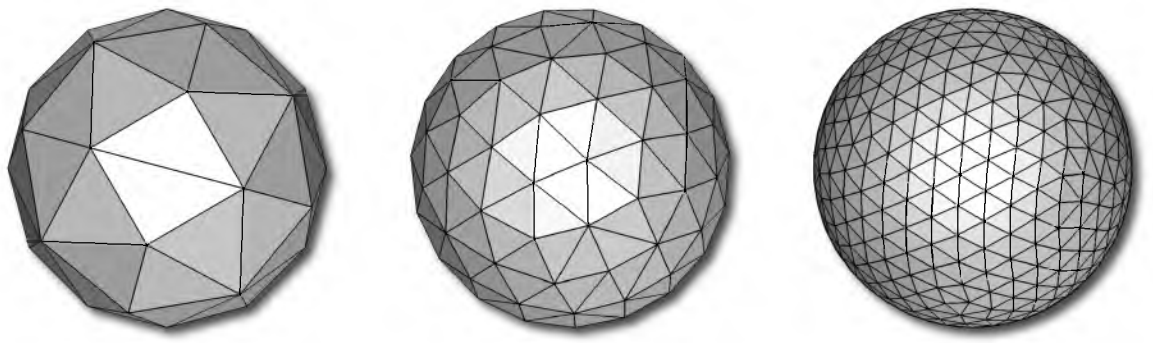


Figure 3.2. Effects of changing curvature scaling parameter s_κ

be equivalent to definitions of local feature size. Figure 3.3 shows an example where the two are equivalent by a factor of one half.

We make the assumption that the topological representation provided as input for the CAD model should be respected in an output mesh. A byproduct of this approach is that some models have adaptivity in regions that are of little benefit to representing the geometry of the model. One could remove adaptivity in regions that do not actually need it by taking a pass over the model and detecting topological junctions that are G^1 continuous, and flagging them to be ignored. The remaining geometrically discontinuous junctions could then be preserved using the sampling scheme.

Gap size is approximated directly from interparticle relationships. Particles store which k-cell they lie on, and each k-cell stores which particles lie on it. We define the topological constraint σ_τ to be the shortest distance from particle p_i to another particle p_j lying on a nonadjacent feature, that is, a 0-cell particle interacting with another 0-cell particle, a 1-cell particle interacting with another 1-cell particle, or a 0-cell particle interacting with a 1-cell particle that does not terminate at that 0-cell. This notion can be extended to 2-cells as well. We further provide a scaling factor s_τ as a user parameter to allow for higher densities of particles within these regions. This proves useful when sampling highly elongated surfaces, with parallel k-cells. Scaling the distance σ_τ allows more rows of particles, allowing for better energy minimization, and ultimately better triangulations.

Figure 3.4 illustrates how changing the topological scaling parameter s_τ effects the

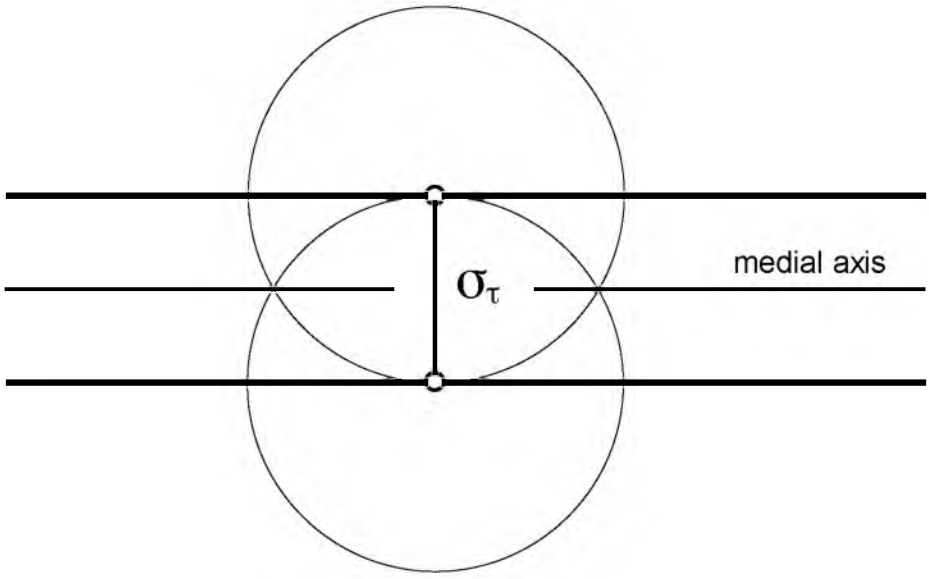


Figure 3.3. Gap size constraint on sizing field. In this case, the gap size is equivalent to the distance to the medial axis by a factor of two.

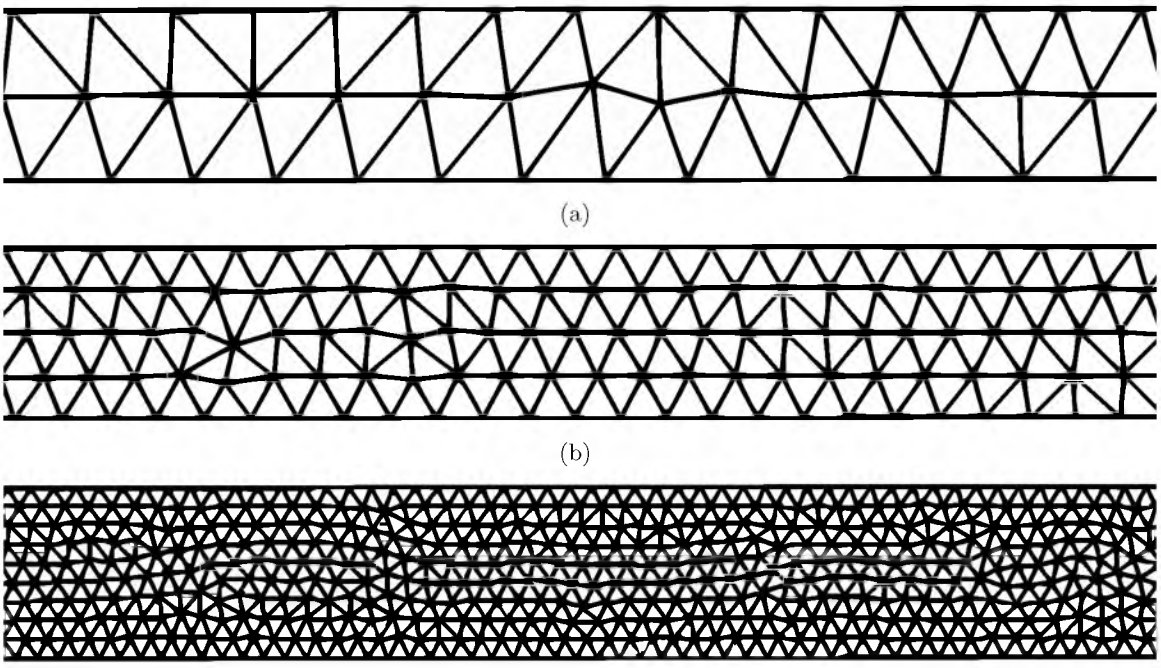


Figure 3.4. Effects of changing topological scaling parameter s_τ

distribution of particles. As the parameter decreases, more particles can fit in the same space enclosed by neighboring 1-cells. Depending on the minimum allowable curvature, and scaling, this parameter may or may not come into play.

3.2.1.3 Lipschitz Continuity

In order to provide finer control over the adaptivity of the particle samples, the system adheres to a Lipschitz constraint $\sigma_{\mathcal{L}}$ that enforces the Lipschitz continuity $\mathcal{L}$ on the sizing field. The Lipschitz condition can be expressed in terms of the formulation as:

$$|\mathbf{x}_i - \mathbf{x}_j| \leq \mathcal{L}\,|\sigma_i - \sigma_j| \tag{3.13}$$

The $\sigma_{\mathcal{L}}$ induced by this constraint is simply the minimum allowable value that satisfies this condition:

$$\sigma_{\mathcal{L}} = \min_{j \in N} \{|r_{ij}|\,\mathcal{L} + \sigma_j\} \tag{3.14}$$

Respecting this Lipschitz continuity provides more gradual adaptivity between areas of high and low particle densities. Lower values of $\mathcal{L}$ produce samplings that result in more isotropic triangles, while large values provide increased levels of adaptivity, at the cost of isotropy. Figure 3.5 illustrates the effects of changing the Lipschitz parameter, $\mathcal{L}$, for a conic surface patch. The parameter is set values ranging from 0.0 (a) to 0.3 (d). When $\mathcal{L}$

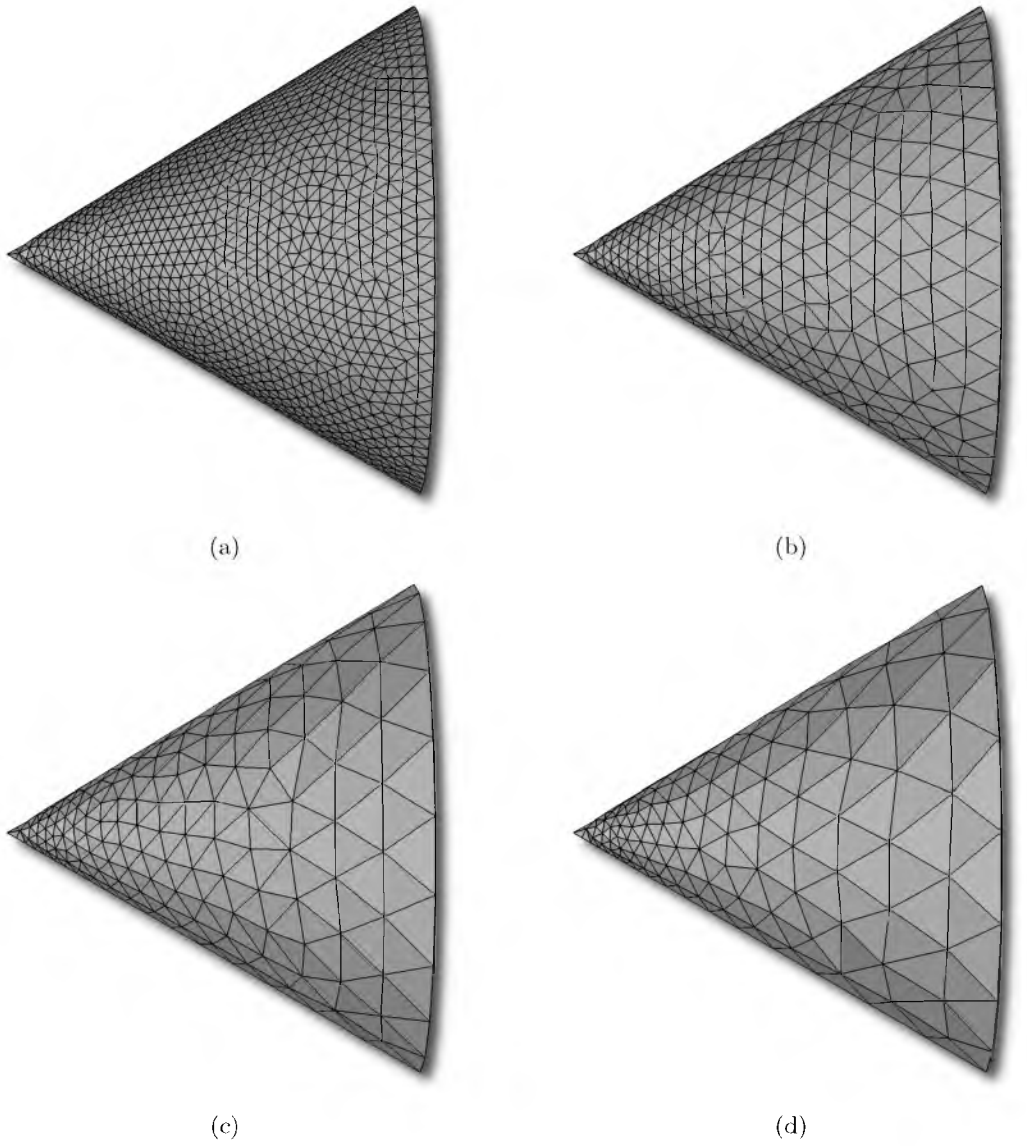


Figure 3.5. Effects of changing Lipschitz parameter $\mathcal{L} = 0.0$ to $\mathcal{L} = 0.3$, (a) to (d), respectively.

goes to zero, a uniform sizing field is produced, fitting the smallest constraint on the model. In this case, the sharp tip of the cone is the limiting feature. We found a default value of 0.3 provides a good trade-off between triangle quality and adaptivity.

It is worth noting that the Lipschitz continuity is not satisfiable for arbitrary surfaces. Because we place samples hierarchically, it is possible the sizing field may need to adapt more quickly on the interior of the surface than it does on the edges. In these situations, the Lipschitz constraint needs to be relaxed to allow the sizing field to adjust.

3.3 Algorithm

The implementation takes as input a parametric model and outputs a triangular mesh. We use the middleware package CAPRI [52] to provide us direct geometry access to shapes generated by CAD software. It also gives access to the topology of the model, including access to the 0-cells, 1-cells, and 2-cells, and their boundary adjacencies. In this section, we elaborate only on the parts of the update algorithm that are independent from the middleware.

3.3.1 Particle Optimization

The sampling algorithm consists of three phases: Phase 1 optimizes 0-cell and 1-cell samples based strictly on the curvature and the Lipschitz constraints, σ_κ and $\sigma_\mathcal{L}$. Phase 2 continues the 0-cell/1-cell optimization, but includes the topological constraint σ_τ. Finally, Phase 3 optimizes samples with surface patches. A phase is considered complete when the change from one iteration to the next drops below some threshold.

We initialize Phase 1 by placing one particle on each 0-cell, and one particle on the midpoint of each 1-cell. Along the 1-cells, splitting happens to increase particle density as the sizing field is inferred. Similarly, if user parameters make any 1-cell particle unnecessary, it will be deleted. Phase 3 is initialized by placing k random samples in the parameter domain of the surface. Each iteration of the optimization, a particle updates both its position as well as its sizing field value σ_i. A scaling factor λ_i is used to increase stability. Pseudocode for the updates of particle positions is shown in Algorithm 1.

Algorithm 1 Position Update

1: **for all** particles **do**
2: Compute energies E_i, dE_i (Equations 3.1,3.2)
3: Compute velocity $\mathbf{v}_i^{xyz}$ (Equation 3.7)
4: Transform to parameter space, obtain $\mathbf{v}_i^*$ (Equation 3.8)
5: Compute scaling $\mathbf{v}_i^{*\text{new}} = \lambda_i \mathbf{v}_i^*$
6: Compute new particle position $\mathbf{u}_i^{\text{new}}$ (Equation 3.9)
7: Transform to world space $\mathbf{x}_i^{\text{new}}$
8: Compute the new energy value, E_i^{new}
9: **if** $E_i^{\text{new}} >= E_i$ **then**
10: **if** $\lambda_i <= \lambda_{\min}$ **then**
11: Skip to next particle on list
12: **end if**
13: Decrease λ_i by a factor of 10 and go back to Step 3.
14: **end if**
15: **end for**

3.3.2 Corner Cases

The motivation for splitting the optimization of 0-cells and 1-cells into two phases is illustrated in Figure 3.6. When it comes to enforcing the topological condition, just as feature size goes to zero in discontinuous corners, so does the notion of topological feature size. Left unchecked, particles in corners would continually shrink their σ_τ, split, and move in closer to the corner.

To curtail this response, we detect and label corners in the first phase. Figure 3.6(a) shows what one corner might look like after Phase 1 has completed. Notice only the curvature and Lipschitz constraints have been met. The σ_i value of the particle on the 0-cell is saved as the size of the 0-cell's corner ball. This is similar to the protecting ball idea in Delaunay meshing [29]. Figure 3.6(b) shows the same corner once Phase 2 has completed. The topological constraint is satisfied for all particles that lie outside of the corner ball. The particles inside adapt smoothly and guarantee the sampling terminates. An alternative approach would be to fix the position of particles laying on this corner ball boundary. The downside to such an approach is that it could easily violate the Lipschitz constraint. With corner cases considered as part of the σ_i constraint, the pseudocode for the sigma update is shown in Algorithm 2.

3.4 Parameter Space Triangulation

The proposed formulation builds a distribution of samples in 3D. To construct a mesh from these samples, one alternative would be to directly build a 3D Delaunay triangulation of the point cloud. Through pruning and filtration, one could construct the surface triangulation and interior tetrahedralization. However, because of the parametric nature of the system, we can instead construct triangulations for each 2-cell and its boundary

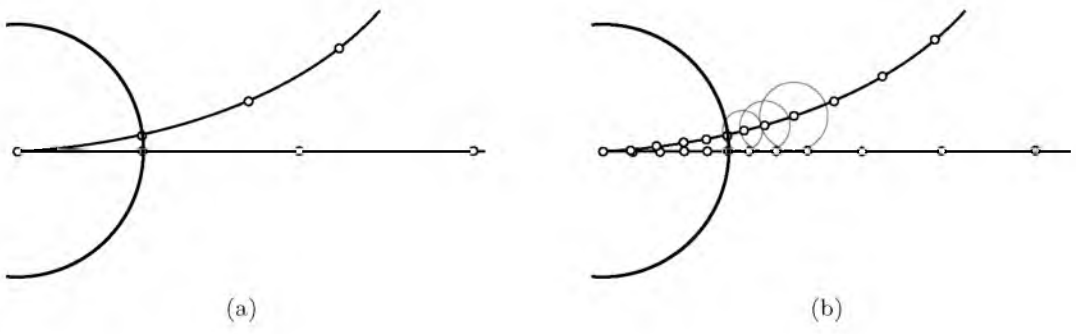


Figure 3.6. Corners are sampled using a two-phase strategy. (a) Phase 1, respecting only curvature and Lipschitz constraints. (b) Phase 2, additionally respecting topology constraints outside the corner ball.

Algorithm 2 Sigma Update

for all particles **do**
  if $p_i \in$ 1-cell **then**
    for all $p_j \in N$ **do**
      if edge(p_i) $\neq$ edge(p_j) and not in corner **then**
        Set topology constraint $\sigma_\tau = \min\{\sigma_\tau, |x_i - x_j|\}$
      end if
      Set Lipschitz constraint $\sigma_\mathcal{L} = \min\{\sigma_\mathcal{L}, |r_{ij}|\,\mathcal{L} + \sigma_j\}$
    end for
  end if
  Satisfy Eq. 3.10: $\sigma_i = \max\{\sigma_{\min}, \min\{\sigma_{\max}, s_\kappa\sigma_\kappa, s_\tau\sigma_\tau, \sigma_\mathcal{L}\}\}$
end for

in the parameter space. This dimensional reduction gains us speed in terms of building the triangulation. Still, particles distributed well in 3D may be in poor configurations in their corresponding parameter space; we account for this using local modifications after constructing an initial triangulation.

Because the parameter space set of samples may be highly distorted, we first perform an affine scaling to regularize the 2-cell as much as possible. Figure 3.7 shows an example of one such scaling, transforming the parameter space 3.7(a) into parameter space 3.7(b). We obtain this transform by solving the least squares solution to the transform that best preserves Euclidean distances. This constraint can be expressed as:

$$\mathbf{A}\,|\mathbf{u}_i - \mathbf{u}_j| = |r_{ij}| \tag{3.15}$$

This constraint urges pairwise distances between particles' 2D parametric coordinates to be as similar as possible to pairwise distances in 3D euclidean space. While we found this transform to be sufficient for the models we tested, clearly more sophisticated transformations could be utilized to remove distortion more uniformly in parameter space. It is conceivable that a sufficiently distorted surface patch might not benefit from a transformation as simple as an affine scaling. Given that such a system already requires the ability to query the surface patch, an optimal transformation would probably need to take surface information into account as much as the world space position of particle samples.

Next, for each 2-cell, we construct the 2D Delaunay triangulation of its particle samples as well as the samples on its boundary curves using Triangle [95]. This triangulation has two problems which we address. (1) This triangulation includes extra triangles (within the convex hull) that may in fact be trimmed portions of the parameter space. (2) The quality of the triangles lifted back in 3D may be poor.

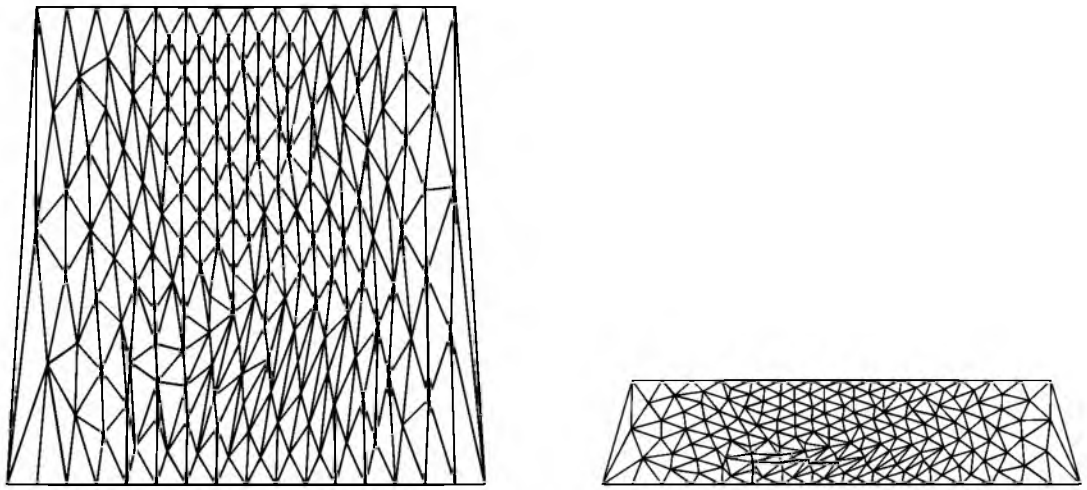


Figure 3.7. Example parameter space before (a) and after (b) affine scaling.

The hierarchical sampling scheme is devised in part to correct for the first concern. The samples of the 1-cells create a dense sample of each curve in both spaces. Moreover, the particle energies on these samples repel particles within neighboring 2-cells away. As a result, these samples act in a role similar to a weighted sample used in recent Delaunay refinement schemes [26]. If each curve is sampled densely enough such that in the 2D triangulation each pair of adjacent 1-cell particles has a Delaunay edge, then we can recover the 1-cell. While an explicit proof is out of the scope of this work, we note that the experiments indicate we can handle arbitrarily sharp edges, without the need for a weighted Delaunay. If we were using a full 3D Delaunay triangulation without weights, we would suffer from angle limitations, as noted by [85].

Having the 1-cells recovered as a sequence of edges in the 2D Delaunay triangulation is sufficient to prune away triangles that are exterior to trims. Once we have pruned these triangles, the remaining triangles are lifted to form the surface triangles of the 2-cell in 3D. However, because of the distortion of the 2-cells parameterization, they may be of poor surface quality. A recent result of Cheng and Dey [23] discusses a scheme to use edge flipping to recover Delaunay surface triangulations. A *Gabriel* property is enforced for each triangle, requiring that each triangle's diametric ball be empty (a stronger version of the Delaunay property). We use a similar scheme; for each edge, we check if two triangles that share that edge have diametric balls that do not contain the opposite, unshared vertex. If they do not, we flip. The recent theoretical result of Cheng and Dey showed this property would only work for ϵ-dense surface triangulations; however, we found the point sets to

be flippable in nearly all cases. A few rare edges could flip indefinitely. To break these ties, we selected the triangle pair that maximize the minimum circumradius (similar to the Delaunay property). Figure 3.8 shows an example of a two-patch sphere model that undergoes this process. Figure 3.8(a) shows the mesh resulting from the affine scaled 2D Delaunay triangulation, while Figure 3.8(b) shows the mesh after edge flipping.

3.4.1 Tetrahedralization

The resultant triangulations are not true 3D Delaunay as we do not ensure that each triangle has a circumball empty of all other points. However, we found they still had two desirable properties. First, nearly all triangles had excellent aspect ratio (shown in the experimental results). Second, these meshes were quite suitable for a constrained Delaunay triangulation that preserves each triangle. We use TetGen [101] to generate high-quality tetrahedralizations of these surface meshes.

3.5 Evaluation

We break the evaluation of this meshing technique into two parts. First, we compare it with two other popular triangular meshing techniques for this domain. Then, we evaluate the technique for its own sake, including strengths, weaknesses, and convergence properties.

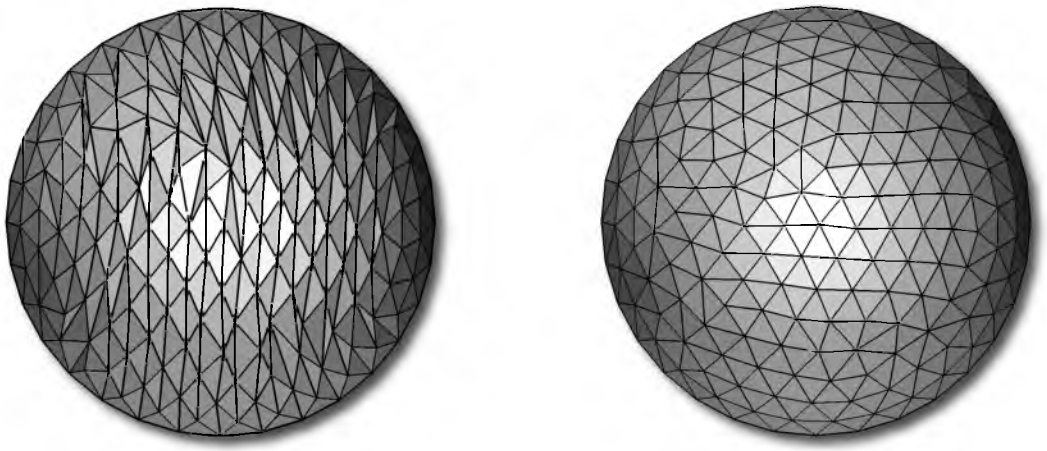


Figure 3.8. Parametric Delaunay before (a) and after (b) edge flips.

3.5.1 Method Comparison

We compare the proposed particle system technique (PSYS) to DelPSC [25] and CAPRI's meshing algorithm [51]. We chose these methods because they were both readily available, and actively used in practice. We evaluate the three methods using surface triangle aspect ratio and volume tetrahedra aspect ratio. To provide a fair test environment, we hand tuned the parameters of each algorithm to generate surface meshes of approximately the same number of vertices. PSYS uses default settings of $s_\kappa = 2$, $s_\tau = 0.5$, and $\mathcal{L} = 0.3$ for all input models.

Figures 3.9-3.14 show comparisons of the surface meshes for each of the three algorithms. In the insets of these figures, we show close up views of each mesh to highlight how PSYS's adaptivity can build superior geometric approximations using the same number of vertices. While the shape of elements is good for all meshes, PSYS can produce especially isostropic triangles. Even in areas of high variability for curvature, PSYS was able to adapt especially well without sacrificing element quality.

To investigate this aspect further, we report the geometric quality of elements on both the surface triangulation as well as the volume tetrahedra. We use the aspect ratio (circumradius to shortest edge length ratio) as a criterion for mesh element quality. Figure 3.15 shows plots of both mesh quality statistics for the mesh of each model using each algorithm. For triangle quality, in Figure 3.15(a), it is interesting to note that PSYS did exceptionally well in the median case. DelPSC has a user parameter to bound triangle quality; the conservative theoretical bounds to guarantee termination require it to be set near 1.0. In addition, DelPSC does not improve element quality near sharp features. As a result, it outperforms CAPRI's surface meshing scheme (which has no refinement parameters for triangle quality), but its median behavior is slightly worse than PSYS.

For volume meshing, the algorithms all behave quite similarly in the median case as shown by Figure 3.15(b). Because TetGen is used for two of the algorithms, this is not an unexpected result. The full 3D Delaunay refinement used by DelPSC also achieves results on par with the other algorithms. We remark that setting naïve parameters to CAPRI's meshing algorithm would build surfaces meshes not suitable to TetGen. Because CAPRI provides no direct control over the quality of surface triangles, if their angles are too sharp, TetGen's refinement could require an impossible number of insertions. We found that PSYS's good quality triangles always lead to suitable inputs for TetGen.

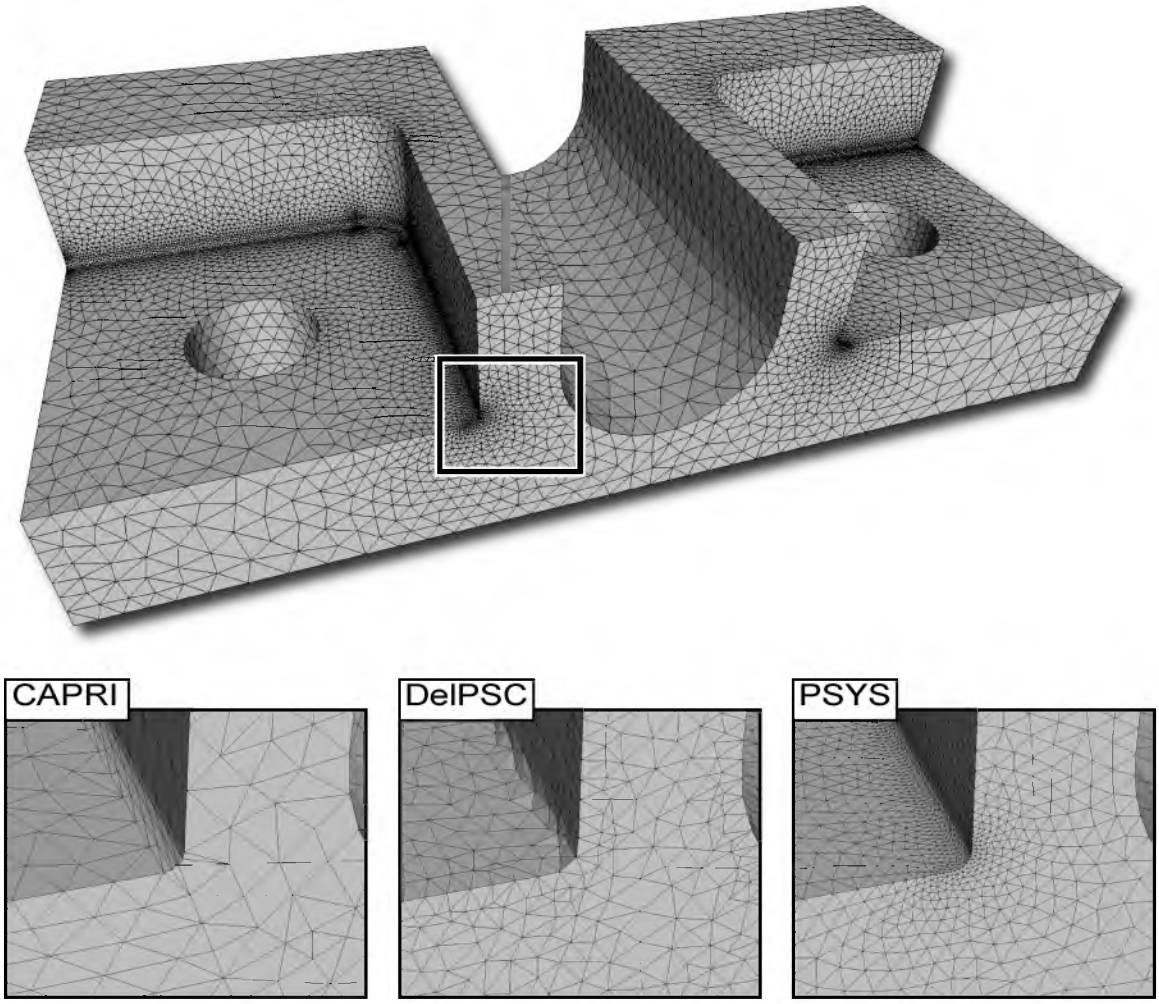


Figure 3.9. The output volume mesh of an engine block (above). Below, a closeup of the mesh is compared against two alternative meshing implementations: CAPRI (left) and DelPSC (center).

3.5.2 Evaluating PSYS

For most models, we are able to obtain good distributions in only a few hundred iterations total. The convergence rates for the particle system to find optimal distributions are based primarily on the number of particles needed and the level of adaptivity. Thus, most iterations take place in Phase 3 of the algorithm. Because particles only interact within a local neighborhood, as the number of samples increases, so does the number of iterations necessary for constraints to travel and allow the system to reach an equilibrium. How adaptivity comes into play is more subtle. We enforce the Lipschitz condition at every iteration, which means boundary values pull down local σ values very quickly. This change propagates outward to areas that can otherwise tolerate a larger σ. This means surfaces

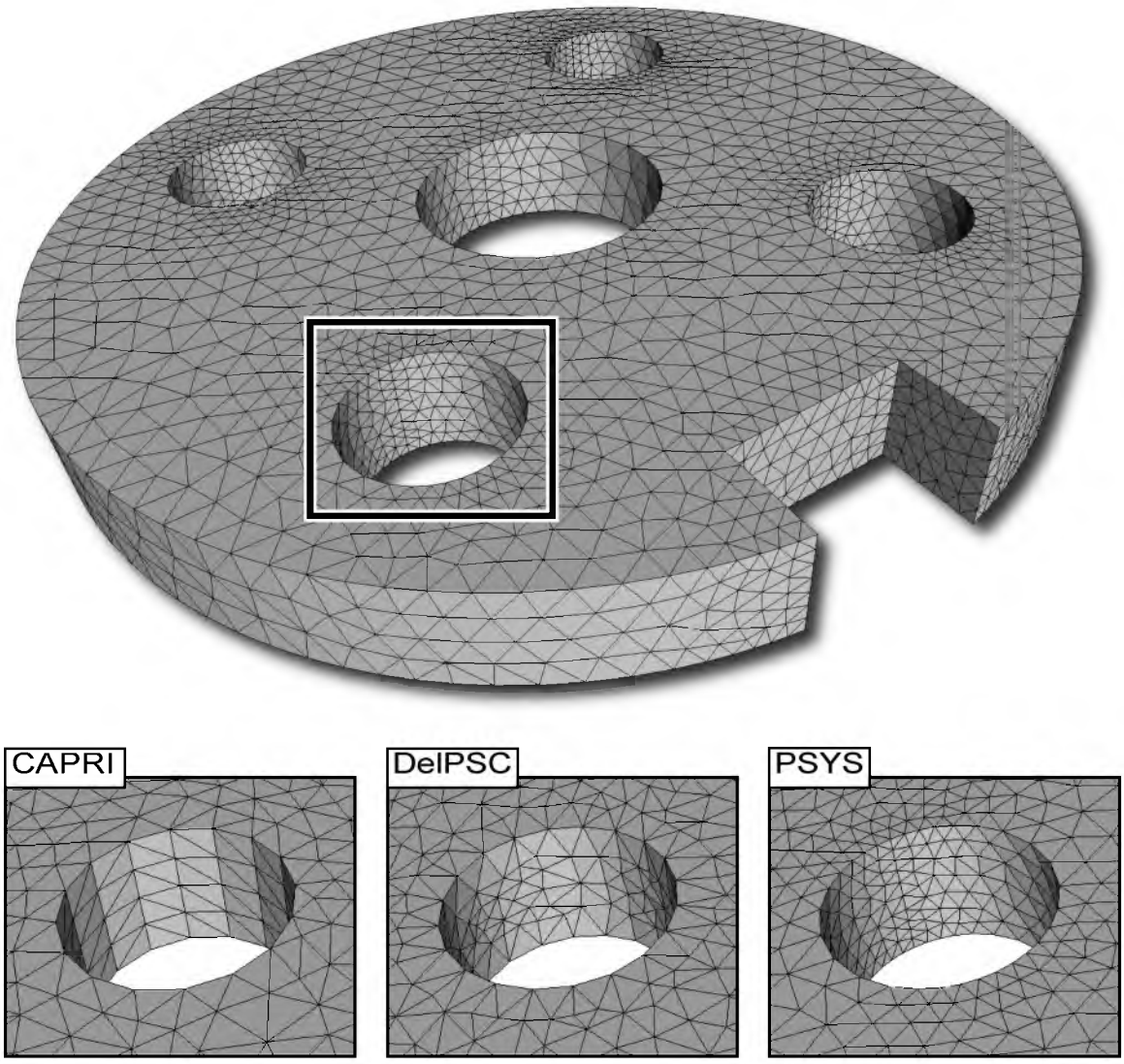


Figure 3.10. The output volume mesh of a disk (above). Below, a closeup of the mesh is compared against two alternative meshing implementations: CAPRI (left) and DelPSC (center).

may become oversampled prior to fitting the Lipschitz continuous field. As the field values increase, so do energies, and particles begin to delete to make room for particles of larger σ values. Relaxing the Lipschitz condition towards the beginning of the energy minimization could provide improved converge rates. Additionally, relaxing the energy requirements for insertion and deletion can improve convergence rates, but at the cost of less ideal packings.

3.5.2.1 Shortcomings

For test examples, we used default settings that all derived in intuitive ways. The curvature parameter was in unit proportion to surface curvature. The Lipschitz parameter

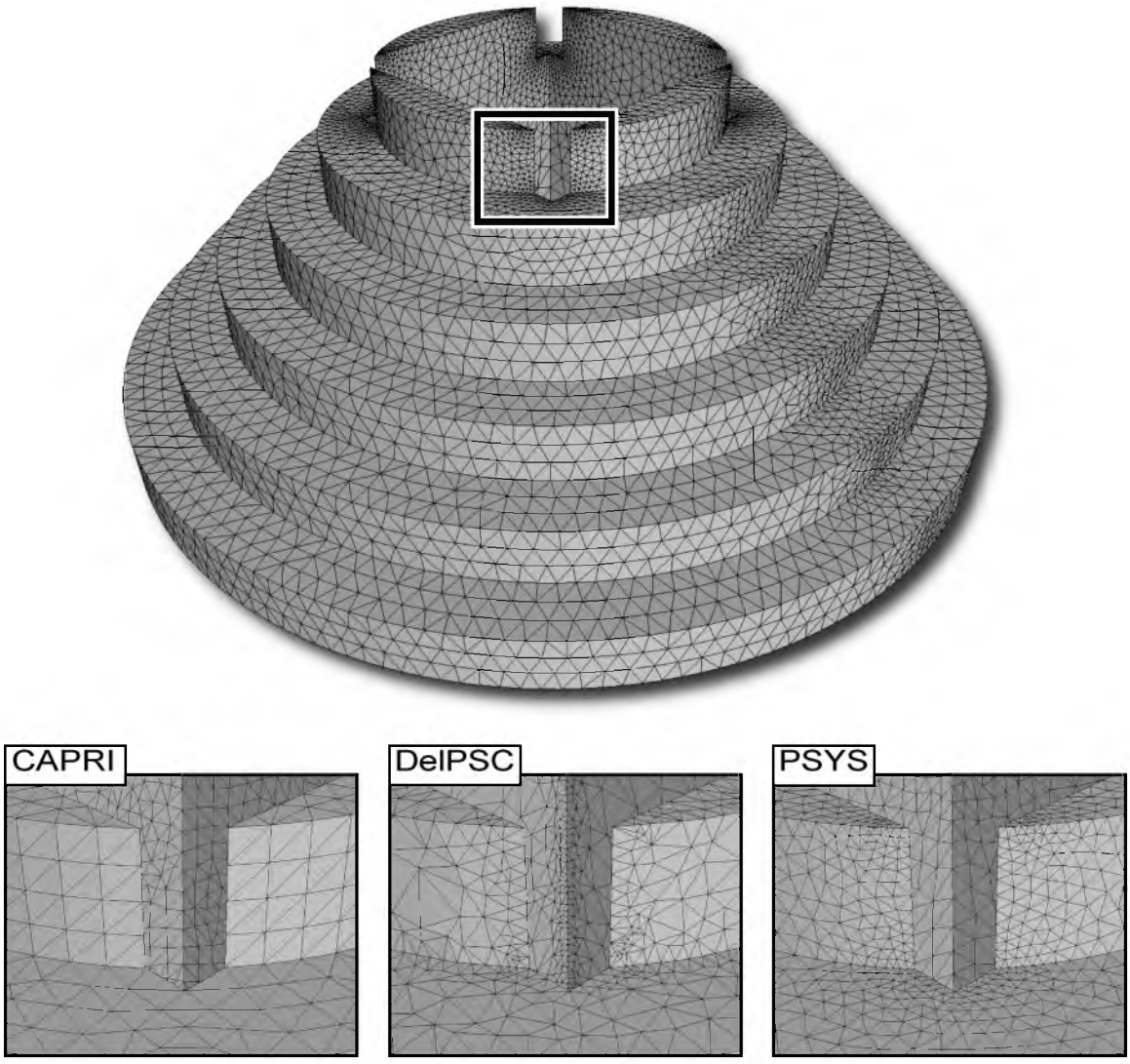


Figure 3.11. The output volume mesh of a hanoi tower (above). Below, a closeup of the mesh is compared against two alternative meshing implementations: CAPRI (left) and DelPSC (center).

provided a fair compromise between triangle isotropy and adaptivity. Even the topological parameter was set to be half unit distance, to ensure sampling room on surfaces with restrictive boundaries. While a typical user might find themselves modifying these values for particular applications, care should be taken to ensure that these values are compatible with the geometry and topology of the model. This is especially true for the topological parameter. For example, the effect of using too large a value is shown in Figure 3.16.

The mesh in Figure 3.16(a), while still valid, has low-quality elements on this edge region, where particles did not have enough freedom to distribute evenly. The mesh in Figure 3.16(b) instead uses the default topological scaling value, and allows sufficient

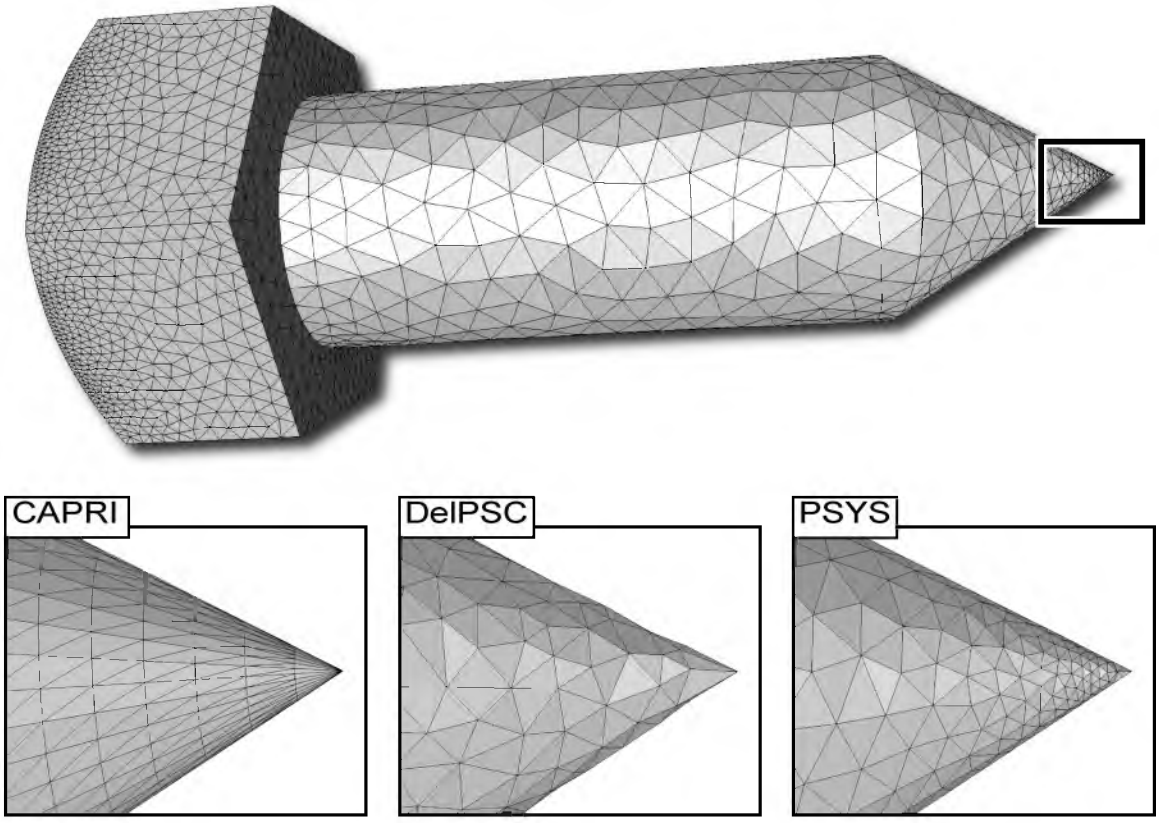


Figure 3.12. The output volume mesh of a screw (above). Below, a closeup of the mesh is compared against two alternative meshing implementations: CAPRI (left) and DelPSC (center).

freedom for particles to distribute and provide a more isotropic triangulation.

Due to the nature of discrete samplings in real spaces, there will always be situations in which a uniform packing is simply not possible. Most often this error will be distributed over much of the samples and will not be visually noticeable. However, occasionally, pockets of low energy show up that cannot be rectified using the proposed rules, because no single particle has an energy that is too low or too high.

Figure 3.17 illustrates this problem on the endcap of a cylinder. The triangulation generated in Figure 3.17(a) has formed an energy pocket in the center, due to the geometry of the cap. The second triangulation in Figure 3.17(b) has no such issue, simply due to a differing particle count. This problem is not unique to PSYS, but rather a shortcoming of formulating a discrete sampling scheme as a continuous energy problem. The likelihood of this artifact occurring could be reduced by increasing the sensitivity of energy measures for splits and deletes, but the gains would be balanced by a decrease in stabilty of the system.

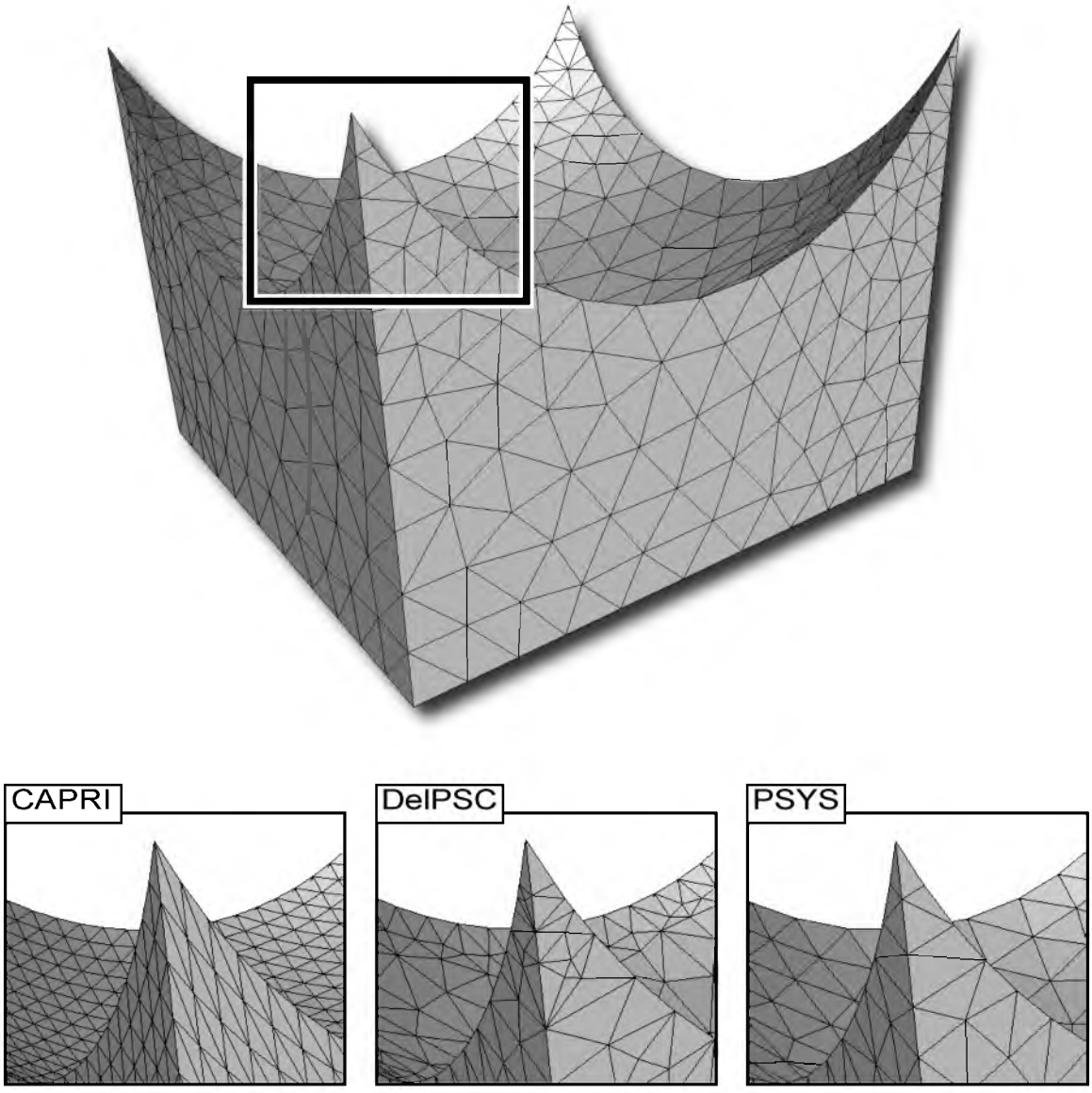


Figure 3.13. The output volume mesh of a table (above). Below, a closeup of the mesh is compared against two alternative meshing implementations: CAPRI (left) and DelPSC (center).

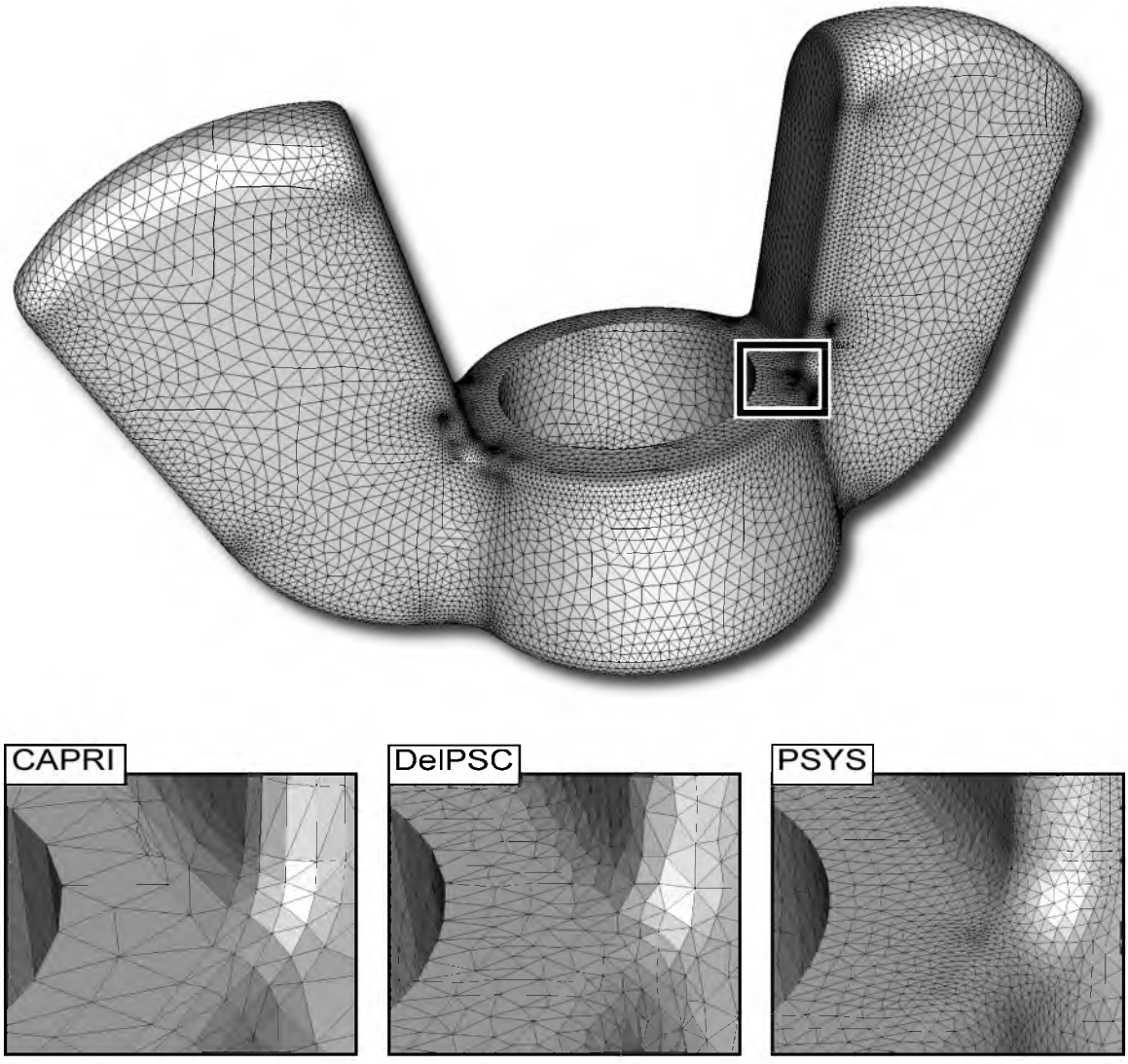
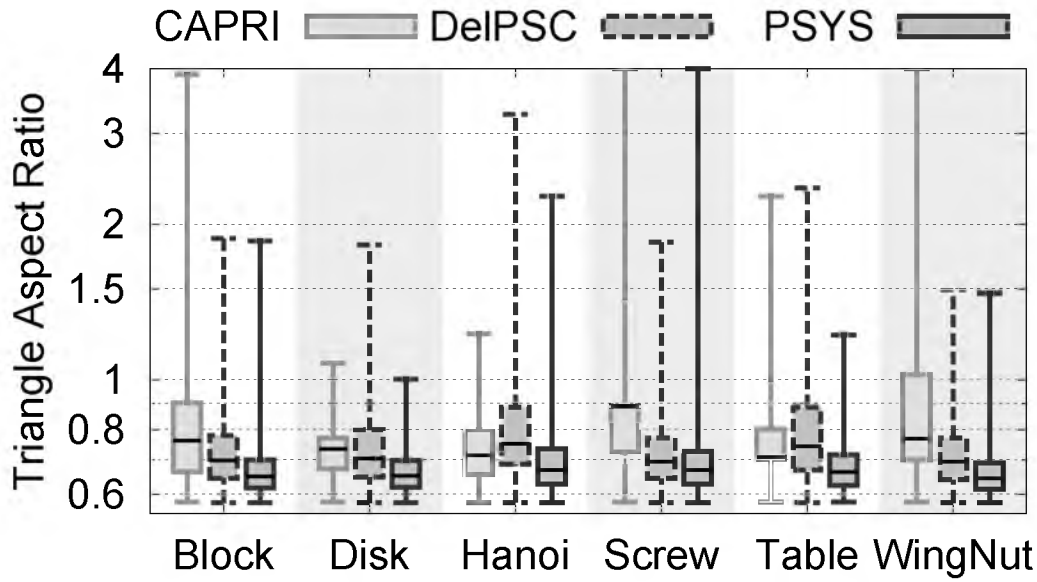


Figure 3.14. The output volume mesh of a screw (above). Below, a closeup of the mesh is compared against two alternative meshing implementations: CAPRI (left) and DelPSC (center).

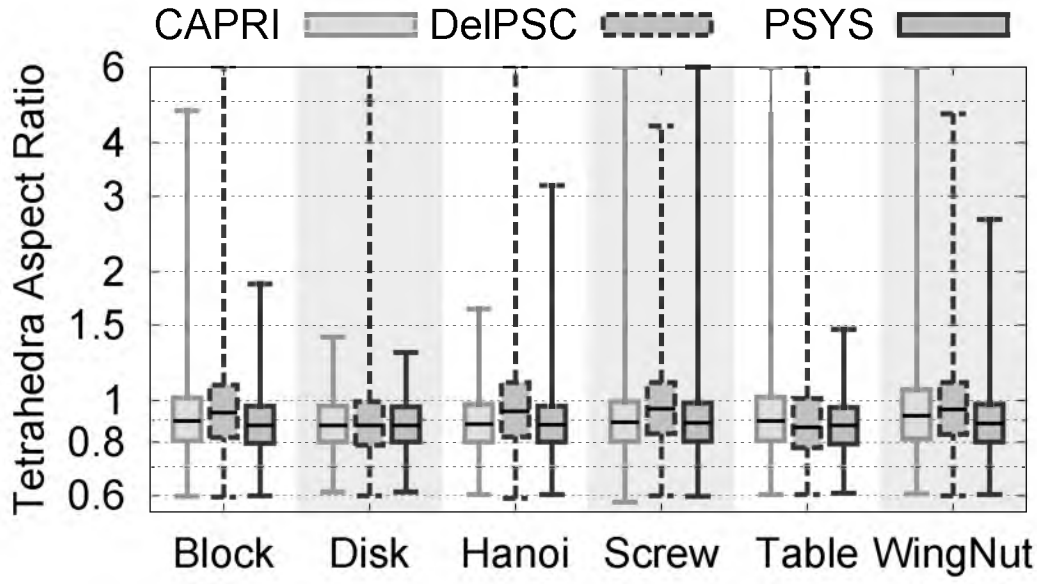
3.6 Discussion

The high-quality results generated from the algorithm in this chapter illustrate how well-suited such particle systems are for sampling parametric domains (Figure 3.18). Compared to the other methods we evaluated, the system was able to generate better quality triangle meshes with the added benefit of adaptively sized elements. Moreover, the sizing field we adapt to can be inferred directly from the point samples, removing the need for model preprocessing.

The success of this technique indicates that there are many unexplored avenues to take with respect to particle meshing. The approach in this chapter is centered around generating



(a)



(b)

Figure 3.15. Box plots of the aspect ratios (circumradius/shortest edge length) on a log scale. We show for triangles (a) and tetrahedra (b) of each output mesh of each algorithm. These plots show the minimum, 25th percentile, median, 75th percentile, and maximum aspect ratio over all elements in each mesh.

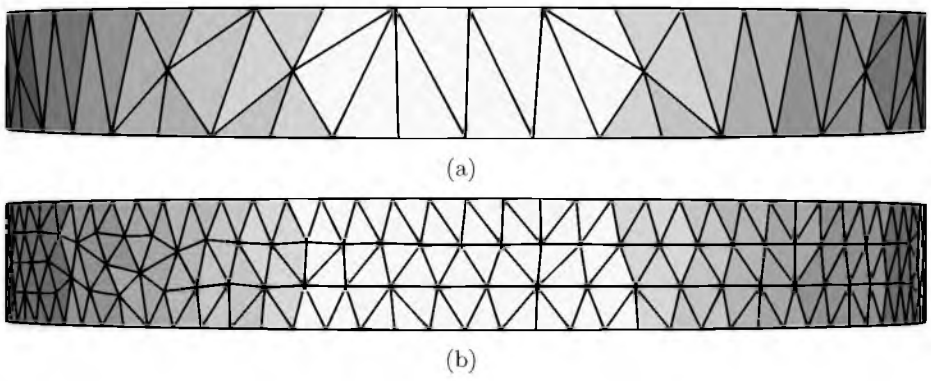


Figure 3.16. Example of bad and good topological scaling parameters: (a) $s_\tau = 1.0$ (b) $s_\tau = 0.5$

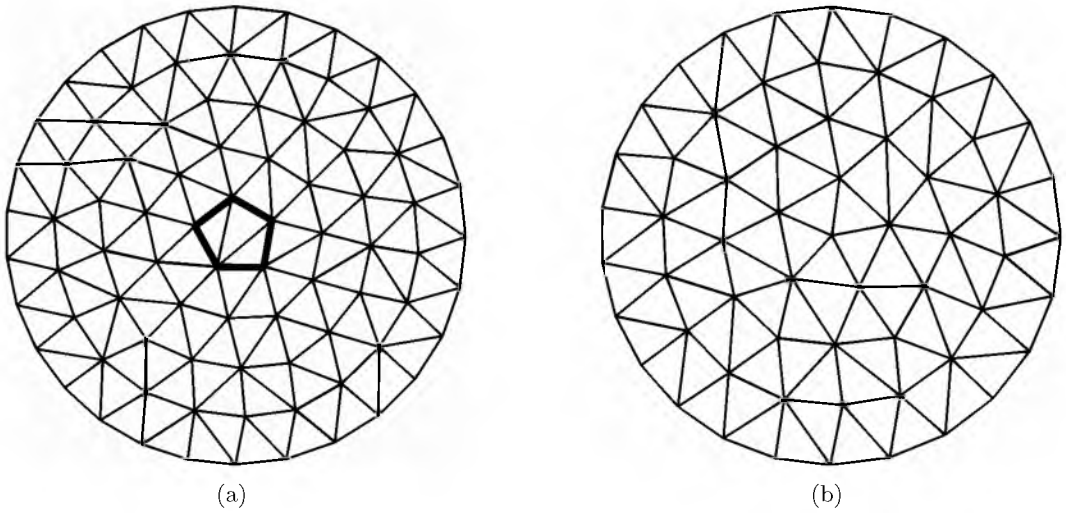


Figure 3.17. Example of a low energy pocket compared to a desired local minimum. (a) Configuration where pentagonal region remains open. (b) Configuration where pentagonal region is completed.

quality isotropic surface meshes, which happen to be good inputs to a constrained 3D Delaunay solution. However, optimizing a particle system directly in 3D space from the start may allow for high-quality, isotropic tetrahedral meshes similar to other variational techniques [1]. An interesting direction would be to infer the tetrahedralization without requiring computing a 3D Delaunay triangulation.

Another avenue we believe could prove fruitful is the introduction of anisotropic kernels to the energy formulation. Doing so could provide an easy and powerful method for generating anisotropic surface and volume meshes. Coupled with adaptivity, these meshes could provide ideal inputs to simulations across many application areas.

The work in this chapter was motivated by quality, and the implementation is not

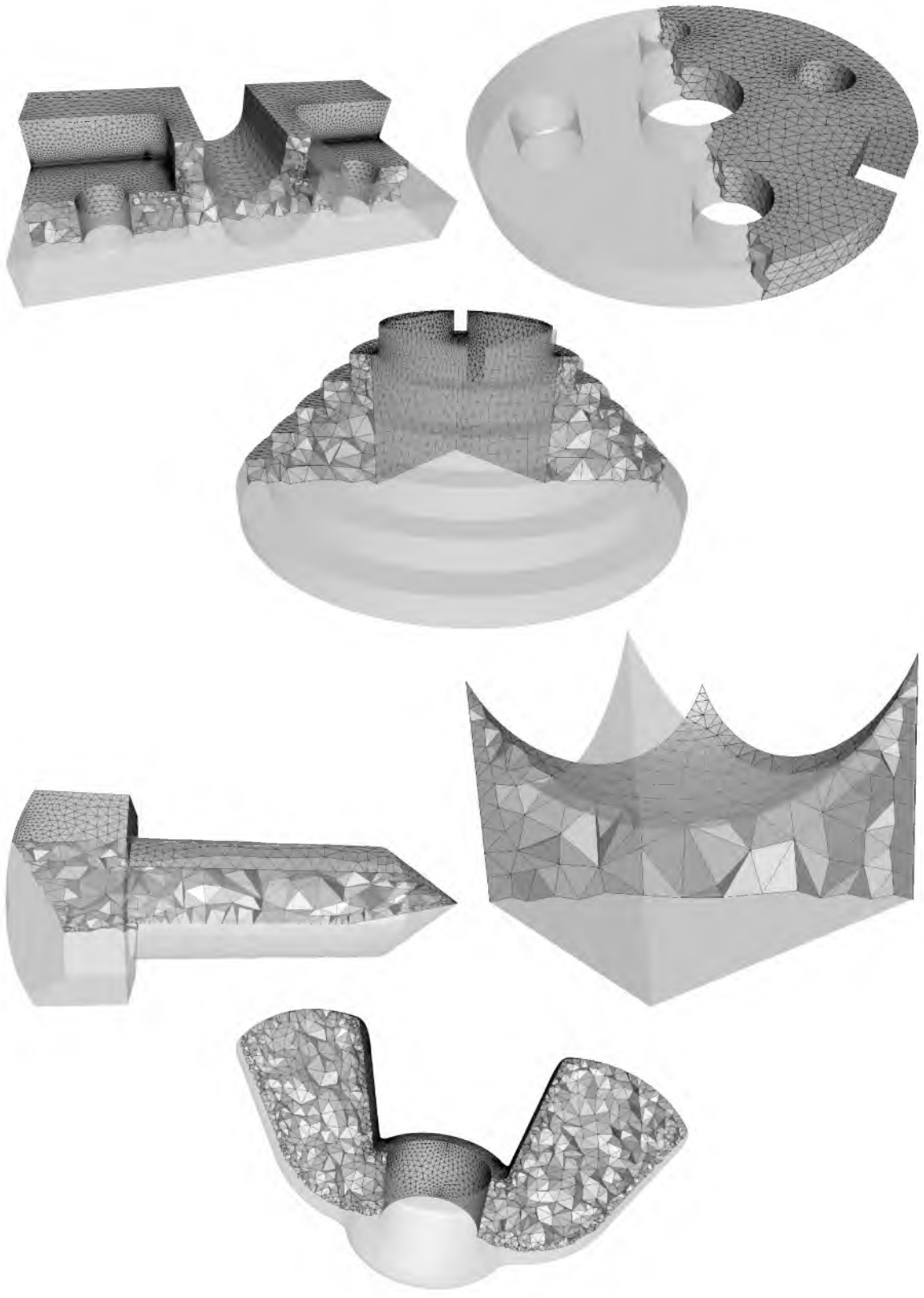


Figure 3.18. The output volume meshes of PSYS. From left to right, top to bottom: the Block, Disk, Hanoi, Screw, Table, and Wingnut models.

optimized for speed. There have been various acceleration strategies for other particle systems that can greatly reduce the running times. Recent work in GPU algorithms [42] have shown that n-body systems can achieve massive performance gains over what would otherwise be an $O(n^2)$ update algorithm.

CHAPTER 4

LATTICE CLEAVING

In this chapter, we describe a new meshing algorithm, *lattice cleaving*, for generating tetrahedral meshes for multimaterial domains that are specified *volumetrically*, as a collection of continuous indicator functions. That is, the physical materials are given by functions on the domain that evaluate to the appropriate material at a given location, and material interfaces are where these functions transition from one value to another. This volumetric specification is natural in biomedical simulations based on images [81], where material boundaries are derived from segmentations or labels, as well as simulations that rely on implicit representations of physical interfaces [30].

The proposed method allows for an arbitrary number of materials, produces high-quality tetrahedral meshes with upper and lower bounds on dihedral angles, and guarantees a degree of geometric fidelity. The output meshes conform approximately to interfaces between materials, including non-manifold regions where multiple materials meet. Moreover, the method is combinatoric so its implementation enables rapid mesh construction. These meshes are structured in a way that also allows grading, in order to reduce element counts in regions of homogeneity.

Lattice cleaving relies on a regular background lattice, with a resolution determined by the user, which is subdivided or *cleaved* to conform to material boundaries. For each cleaved background tetrahedron, it applies and modifies a stencil, used to approximate the geometry while not destroying the good quality of elements in the background lattice. Lattice cleaving requires a small, fixed number of passes through the background grid and therefore leads to reliably fast run times. Results on biomedical volumes and fluid simulations demonstrate the algorithm reliably achieves fast run times, geometric fidelity, and good quality elements. Additionally, we provide proofs showing that both element quality and geometric fidelity are bounded using this approach.

4.1 Background

The literature on unstructured 3D mesh generation is vast, partly as a byproduct of the wide utility of these meshes to application areas in science and engineering. Here we focus the discussion along the two major constraints on this meshing problem: (1) producing high-quality elements and (2) conforming to complex surfaces. In addition, we review lattice-based meshing algorithms which use, in part, similar techniques to those presented in this document.

In the past decade, a significant amount of effort has gone into building high-quality surface meshes [48], particularly through Delaunay triangulations [28]. One of the most popular strategies relies on Delaunay refinement [34, 86], which iteratively inserts sample points on the domain boundary until conditions are met for sufficiently capturing both the topology and geometry of surfaces. These surface meshes are typically inputs for conformal tetrahedral meshing algorithms, with further refinements of the volumetric regions. Boissonnat and Oudot [13] and Cheng et al. [27] pioneered the first variants on provable algorithms for performing Delaunay refinement that capture the topology of smooth, surface-boundary constraints. Extending these ideas to more complex, piecewise-smooth, and non-manifold domains followed [26]. However, these algorithms rely on various strategies for *protecting* features on the material boundaries, and the implementations of these schemes are challenging. Thus, simplifying assumptions are required in the protection scheme to make them practical [14, 39, 83].

The local, greedy strategy of Delaunay refinement schemes tend to find suboptimal configurations for vertices. Variational meshing schemes attempt to overcome this limitation by positioning vertices according to some global energy function. These strategies typically decouple, to various degrees, the vertex placement problem from the triangulation/tetrahedralization problem. Meyer et al. [76] use a variational scheme similar with repulsion between particles (points) to sample multimaterial interfaces and then connect these samples using a Delaunay triangulation. Bronson et al. [20] build on this formulation to build highly adaptive surface meshes for CAD geometries, but do not require expensive precomputations. Yan et al. [115] use an energy formulation based on centroidal Voronoi tessellations to drive particle movements. Tournois et al. [106] alternate between Delaunay refinement insertions and vertex optimizations for high-quality meshes for nonsmooth shapes. These types of optimizations are nonlinear and require multiple iterations on gradient-descent-based strategies to find local minima. Thus, they are time consuming, are sensitive to initializations and parameter tuning, and do not provide typical criteria to

establish guarantees on the quality of the output.

While these works represent only a taste of the most recent work in boundary-constraint meshing, they have a number of interesting shared characteristics. First, each algorithm requires at least one expensive computation, either a Delaunay triangulation in 3D, or a vertex optimization, or both (each of which necessitates an $O(n^2)$ computation), while the variational schemes may require multiple iterations on these computations. Second, these algorithms all decouple the surface constraint of boundary conforming from the volumetric quality constraint, by requiring that the algorithm *first* construct a mesh of the boundary and then next construct a volume mesh given this surface mesh as input (*surface first, then volume*). As a result, they often deal with very poor-quality mesh elements, in particular *slivers* or tetrahedra with nearly cocircular vertices [24, 106]. In contrast, the proposed method follows the strategy of *volume first, then surface*, which has the benefit of being more flexible in how we manage the inevitable compromises between geometry and quality.

4.1.1 Tetrahedral Element Quality

While a large number of measurements and ratios are used to judge the quality of elements in meshes, in this work, we focus on isotropic measures of quality applicable to linear finite elements [97]. These qualities, while somewhat generic, have the advantage of being numerically useful for the large class of elliptic operators that appear in FEM simulations of many physical phenomena, such as incompressible flows, diffusion, and electric fields. While there are many reasonable measures of tetrahedral mesh quality, we rely on the worst-case dihedral angles (both minimum and maximum) over all tetrahedra. The distance of dihedral angles from 0° and 180° correlates with most other common element quality measures.

Measures of quality are typically independent of element size. Adaptivity of mesh size/resolution provides another key ingredient in the definition of element quality. Both Delaunay refinement [96] as well as variational schemes [3] have been used to improve and adapt volumetric element quality, as well as more greedy optimizations driven by local mesh improvements [47, 59]. Moreover, isotropic element quality is also indifferent to element orientations; i.e., it penalizes anisotropy in all directions equally. The proposed work provides graded meshes with smaller elements near surfaces, but does not address the problem of adaptivity and anisotropy directly.

4.1.2 Lattice-Based Meshing Approaches

A very common strategy for building meshes is to start with a high-quality (e.g., regular) background mesh and modify it to adhere to geometric constraints. However, the problem of making a regular lattice conform to an arbitrary surface presents some challenges, especially when tetrahedral quality is a concern. One strategy is to cut (or cleave) the cells of the input lattice to match the surface, an idea popularized by the well-known marching cubes algorithm for isosurfacing [69], and the related dual contouring method [56]. Constrained delaunay triangulation [32, 98] can then be applied to generate volume-filling tetrahedra [117]. The different configurations of surface/cell intersections are typically represented by *stencils* with the appropriate topology. Several authors propose surface reconstruction with a piecewise linear approximation of surfaces as they cut through the tetrahedra of a body-centered cubic (BCC) lattice [12, 78], with extensions to non-manifold surfaces using a collection of indicator functions (instead of the single scalar field for isosurfacing). These algorithms examine indicator functions locally at each vertex of the mesh element. Depending on which indicator is maximal, they next label each vertex with a material, a generalization of inside/outside for isosurfacing.

Working with lattices has advantages beyond just surfacing. For instance, an octree can be used on a regular lattice to facilitate adaptively sized elements [116]. Zhang et al. [118] use a regular lattice and encodings to achieve surface simplification while preserving topological features. This work was extended to generate tetrahedral as well as hexahedral elements [119] and within multimaterial domains [121].

Another strategy for conforming is to warp a background lattice so that primitives align with boundaries [49]. Molino et al. [77] use a BCC lattice coupled with a red-green subdivision strategy, which they then optimize to conform to the surface. That work empirically achieves good-quality tetrahedra, albeit with no proof of bounds. Labelle and Shewchuk [62] propose a combination of lattice warping and stenciling, with appropriate rules that decide which combination of strategies to use, based on the input data, in order to ensure good quality. They describe a computer-assisted proof to compute quality bounds for their *isosurface stuffing* algorithm. Wang and Yu [110] employ a similar approach. The proposed method shares several aspects of the Labelle and Shewchuk method. Like their algorithm, we cut a BCC lattice to conform to a boundary mesh, and like their algorithm, we rely on a threshold α to locally warp the lattice to remove short edges and maintain high-quality elements. However, instead of considering a single smooth isosurface, the multimaterial boundary constraints present nonsmooth and non-manifold structures. This

adds considerable complexity to the algorithm, which in the past has only been approached using additional levels of subdivision, such as in Liu *et al.* [65] or dual-contouring [120]. Here we show instead that a carefully designed stencil set combined with appropriate rules for application provides quality guarantees for the resulting tetrahedra.

4.2 Considerations

The proposed tetrahedral meshing algorithm operates on a collection of indicator functions. We sample these functions onto a body-centered cubic (BCC) lattice. Similar to many surfacing and meshing algorithms [50, 69], we rely on a set of stencils that capture local material configurations. We use the strategy of Labelle and Shewchuck [62] to construct a set of rules for each background BCC lattice tetrahedron that switch between two cleaving modes—either deforming the background BCC lattice or splitting the background tetrahedra in order to conform to boundary surfaces.

Within this context, the multimaterial meshing problem presents several important challenges. Unlike the isosurface case, one cannot easily restrict the size of features, because feature size [3] will always go to zero where three or more materials meet. The complexity within each lattice cell is also challenging. Considering only the material labels at vertices, the number of cases is daunting. Furthermore, even if one represents indicator functions along lattice edges as linear, the number of possible interfaces passing through a single edge grows with the number of materials, regardless of the conditions at the vertices. Therefore, geometric and topological approximations are essential.

4.3 Indicator Functions

There are many papers on extensions of implicit surfaces or level sets to multimaterial interfaces. Here we represent multimaterial interfaces using a set of K-smooth, volumetric *indicator functions*, $F = \{f_i | f_i : V \mapsto \Re\}$ [73, 81]. A material label i is assigned to a point $\mathbf{x} \in V$ if (and only if) $f_i(\mathbf{x}) > f_j(\mathbf{x}) \ \forall \ j \neq i$. For any single material j, a continuous, inside-outside function can be constructed as $\tilde{f}_j(\mathbf{x}) = f_j(\mathbf{x}) - \min_{i \neq j}(f_i(\mathbf{x}))$, and the zero functions of various materials will coincide at shared boundaries.

4.4 Background Lattice and Material Interfaces

Stenciling algorithms rely on a set of regular cells. The configuration of the interfaces on these cells are used to generate an index that corresponds to some predefined tessellation. We employ a BCC lattice (Figure 4.1), where each cell is composed of 8 normal or *primal* cubic lattice vertices, plus a 9^{th} *dual* vertex in the center. In addition to the twelve edges of

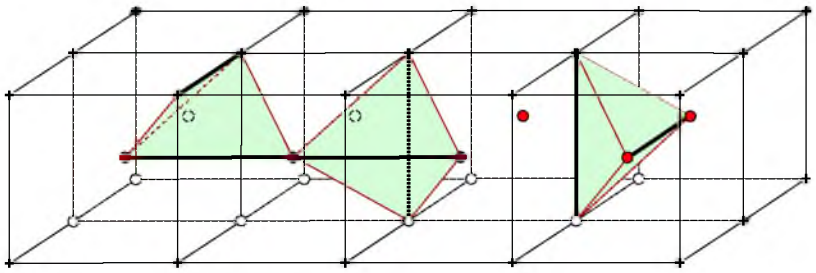


Figure 4.1. The BCC lattice is composed of two grids of primal and dual vertices [62]. Each vertex is incident to 14 edges, 36 faces, and 24 tetrahedra.

a regular cubic cell, there are eight diagonal edges connecting each dual vertex to its cell's primal vertices, and six connecting dual to dual. Fanning out from the dual vertex are 24 lattice tetrahedra, each of which spans two lattice cells. Each lattice tetrahedron is identical in shape, as is each lattice face. The edges connecting both primal and dual vertices differ in length from the edges connecting only primal or only dual vertices. We refer to these edges as long and short, respectively.

For stencils to be applicable, decisions about the structure of each cell must be strictly local and enumerable a priori. The proposed strategy for mapping data onto the lattice entails several approximations. Initially, each lattice vertex represents a single material at that point, which is given by the indicator function with maximum value. If two or more functions are comaximal, the tie is settled by a very small push away from a prioritized (or random) material. Any lattice edge that contains vertices with two different labels contains a material transition, called a *cut-point*, or *cut* [62]. These edge-cuts sample a surface separating two materials.

A similar logic applies to junctions of more than two materials. A lattice face with three unique material labels on its vertices must have an associated transition point where all three materials meet. We refer to this point as a *triple-point* (*triple*). The collection of triple-points in the domain define curves where three materials meet. A lattice tetrahedron may have up to four unique material labels. The 4 vertices, and the four function values associated with the material labels on each vertex, define a single, isolated, material transition point. We refer to this point as a *quadruple-point* (*quadruple*).

We restrict the number of material transitions defined on a tetrahedron. Each lattice simplex may contain at most a single transition point matching its dimensionality: an edge may have only a single cut, each face a single triple, and each tetrahedron a single quadruple. These approximations are the multimaterial generalizations of the approximation that underlies stencil-based isosurface algorithms, which ignore features that pass between vertices.

Figure 4.2 illustrates how such a situation might manifest on an edge. These various material interfaces are defined as the points where the values of indicator functions of the materials on the vertices are equal, and, in general, we assume these locations are given by an *oracle*. In the absence of an oracle, these points can instead be computed, for instance, by a system of linear equations.

For an edge, these transitions lie on the line segment connecting the two vertices. However, for triple- or quadruple-points, they could lie outside of the corresponding triangular face or tetrahedron, respectively. In such cases, these points are projected back onto the tetrahedron, so that local stencils can apply (Figure 4.3). These approximation lead to a smoothing or removal of thin features that fall below the resolution of the grid—i.e., the exceptions to the above conditions are indicative of features that fit between grid points, as proved in Section 4.10.1.

4.5 Stencils and Keys

Selection of the appropriate stencil for any given cell is determined by a key. For the single isosurface case, this key is an ordered concatenation of labels for each vertex of the simplex being stenciled. These labels indicate whether a vertex is inside (-1), outside (+1), or directly on the isosurface (0). If we were to restrict vertex labels to be strictly inside and outside, this leads to 16 configurations, with only eight unique valid cases. Allowing a vertex to sit on the isosurface would lead to 81 possible configurations. In the multimaterial case, this approach to indexing stencils is problematic for several reasons. First, inside of one material is outside of another. This means a label for each material in the volume is needed. The length of the key will need to grow in order to accommodate datasets with larger numbers of materials. Consequently, the number of cases of stencils would need to

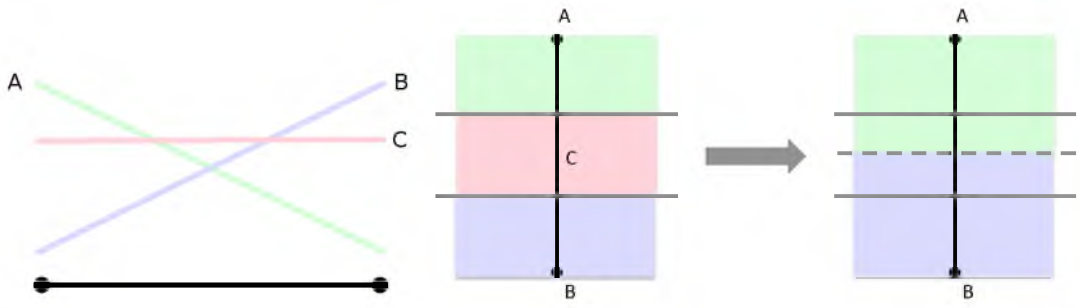


Figure 4.2. An edge with materials a and b maximum on its endpoints, but with a third material c becoming maximum on the interval between.

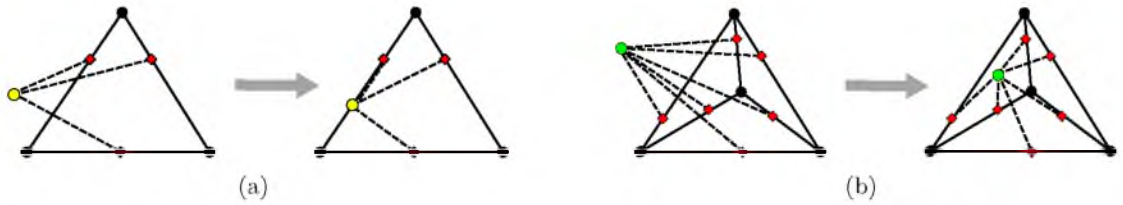


Figure 4.3. Triples (a) and quadruples (b) are forced to lie within the primitive that contains the associated edge-cuts.

fill the space of cases, with many duplicates for similar topologies with differing materials.

One approach to solving this issue is to instead store bits only signifying the number of materials found on a vertex. In this scenario, the case table becomes tractable, but distinguishing element materials during stenciling becomes more difficult. This task requires expensive bookkeeping and may still lead to ambiguous material regions without explicit rules to avoid them.

The solution in the proposed approach bypasses this problem by only defining stencils on lattice tetrahedra with precisely one material per vertex. Intermediate topologies, where multiple materials reside on a single vertex, are generated by various edge collapses on the original stencil. In this way, the challenges of exhaustive case tables and ambiguous regions are avoided. The key for a generic set of stencils in this multimaterial case would be a 6-bit key, each bit indicating whether a particular edge of the lattice tetrahedra contains an edge-cut. This leads to only 64 possible configurations, with many cases being invalid. In Section 4.8, we show that by generalizing all stencils to a single case, we can avoid the need for a lookup table altogether.

4.6 Quality Criteria

Within a lattice tetrahedron, we approximate the material interfaces as a set of triangular facets that connect the various cut-points with the correct topology. With no additional processing of the mesh data, there are only five unique topological cases, distinguishable by the number of edges that contain a cut: 0, 3, 4, 5, or 6. It is impossible for a lattice tetrahedron to contain only one or two edge-cuts. As illustrated in Figure 4.4, these cases are composed of three types of polyhedra: tetrahedra, triangular prisms, and hexahedra.

While these polyhedra admit multiple consistent tessellations, the output tetrahedra could become arbitrarily bad (regardless of the tessellation chosen) depending on where interface points are located. Thus, we define a set of *violation conditions* that characterize the configurations of interface points that lead to bad tetrahedra. These conditions are used

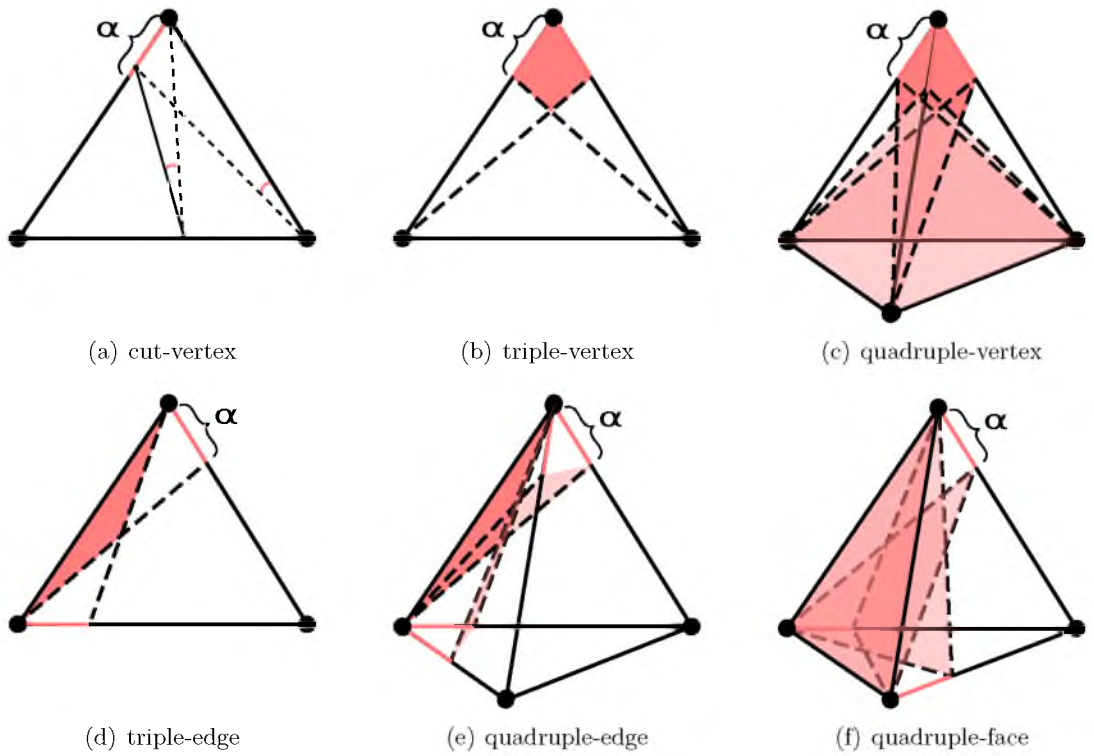
(a) 0-cut (b) 3-cut (c) 4-cut (d) 5-cut (e) 6-cut

Figure 4.4. The five unique interface topologies determined by the number of cut-points present on a lattice tetrahedron.

to decide when it is appropriate to warp the background lattice (changing topology) and when it is appropriate to leave a configuration intact. The conditions entail a threshold on the proximity between features, denoted α, and are expressed as a fraction of the edge lengths on the background lattice.

There are three different ways in which an interface might be *violating*. First, an interface point of any type may violate a lattice vertex. A cut violates a lattice vertex if it lies within a distance α to it along the shared edge. As shown in Figure 4.5(a), even in 2D, no matter how you choose to tessellate a face, there will always be an angle that is arbitrarily bad as the cut approaches the lattice vertex. This principle extends to interface points of higher order (i.e., triples and quadruples). Triple-points can move within the 2D space interior to a lattice face, and so their vertex violation region is a quadrilateral patch. This patch is formed by the intersection of two half spaces. Each halfspace is defined by connecting the point at distance α on one edge, to the opposite lattice vertex (Figure 4.5(b)). Similarly, quadruple-points can be anywhere inside the lattice tetrahedron, so their vertex violation regions are formed by the intersection of three half-spaces defined by planes (Figure 4.5(c)).

The second group of violations pertain to edges. Degenerate tetrahedra can also arise if triple-points or quadruple-points lie too close to an edge. We define the notion of edge violation in a manner consistent with vertex violations, similarly bounding the angles. Dividing lines are formed between each vertex on the edge and the respective α position on the edge opposite that vertex (Figures 4.5(d,e)). Finally, a quadruple-point has one additional violation condition, arising from its distance to adjacent faces. This violation region for faces follows the same logic as the others (Figure 4.5(f)). Three planes are defined using the alpha parameters. Each plane contains two vertices on the face in question, and the alpha position on the edge incident to neither of these two vertices. The intersection of the halfspaces defined by these three planes forms the 3D violation space. With these violation rules in place, the next section provides a set of operations that can remedy any set of interface points that are in a violation condition.



(a) cut-vertex (b) triple-vertex (c) quadruple-vertex

(d) triple-edge (e) quadruple-edge (f) quadruple-face

Figure 4.5. An interface point violates a feature if it falls within an intersection of half spaces defined using α. Vertices can be violated by (a) cuts, (b) triples, and (c) quads. Edges can be violated by (d) triples and (e) quadruples. Faces can only be violated by (f) quadruples.

4.7 Snapping and Warping

The lattice cleaving algorithm uses two fundamental operations to ensure mesh quality. A *snap* operation merges an interface point with another point of lower order, collapsing the implicit edge between them in an output stencil. This operation is performed on interface points that are in violation, ensuring output stencil tetrahedra do not span bad angles. In conjunction, lattice vertices are *warped* spatially in order to conform to the interface surfaces.

The multimaterial cases introduce additional complexity into processing the snaps and warps. In the two-material case, Labelle and Shewchuk [62] warp a lattice vertex to a single violating cut and remove all adjacent cuts, effectively pulling them into the warped vertex. In the multimaterial case, this is unsatisfactory, because the adjacent cuts could be interfaces to several different material sets.

If interfaces of different material sets are violating a lattice vertex, no single warp position can satisfy the surface constraints. Therefore, a number of strategies may be used to choose a suitable warp location. We use the center of mass of these violations, because it is both

easy to compute and distributes error proportionally among each interface. alternative approaches might be to minimize a quadric error term for each surface, or to choose a preferred surface representative.

After snapping and warping a vertex to remove its violations, additional material interfaces may still be present on the incident edges and faces. Because these edges and faces will also be warped by the movement of the lattice vertex, the implicit surfaces may intersect them at new locations. Thus, we must update the locations of the remaining cuts and triples on incident edges appropriately, either by geometric intersection tests or via querying the oracle.

When an edge moves because one of its vertices is warped, any cut on that edge must move along the interface the cut represents. In practice we use a linear approximation to the interface surface to perform this update. If the cut interface is of the same type as the violating cut that caused the warp, it naturally gets pulled into the snap like the two-material case. If the new location of the cut is no longer on the edge (e.g. moves off of one of the ends), we bring it back onto the line segment at the appropriate end point. In this way, the stenciling operation remains local. We call this operation of recomputing the position of an interface on the warped lattice a *projection*, shown in Figure 4.6. If the new cut position is violating, we perform a snap to the lattice vertex, and warp the vertex only if it has not already been warped previously. In this way, each lattice vertex undergoes at most one warp.

If a lattice face moves because one of its vertices is warped, any triple on that face may also move. We update its location using the same strategy as with edges. If the triple leaves

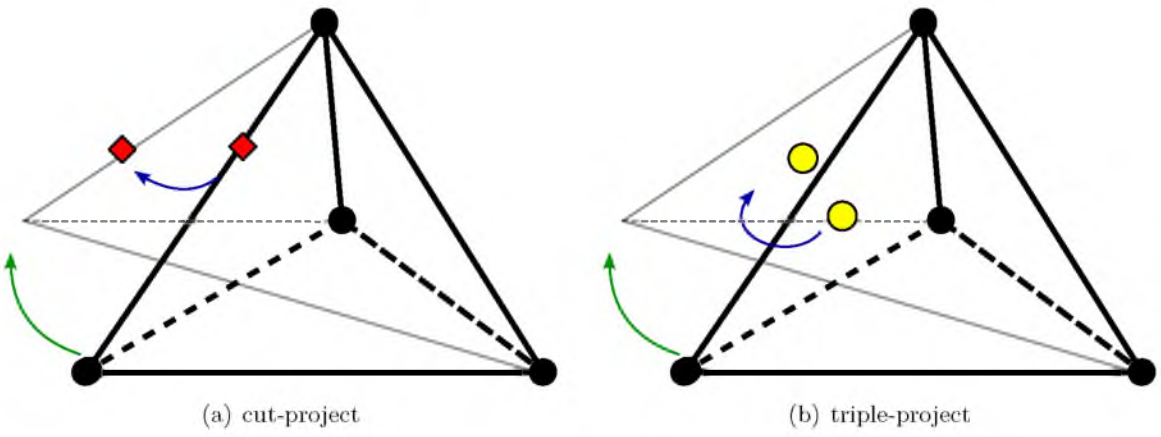


Figure 4.6. When a vertex warps (green arrow), (a) cuts and (b) triples on incident faces must be updated (blue arrow) to reflect the their new locations on the surfaces.

the face, we bring it back on, and follow up with appropriate snaps and warps as needed. Quadruples need no projection unless a face moves in such a way that the quadruple falls outside of the new tetrahedron—in which case it will be moved onto the nearest edge/face and collocated with the corresponding cut/triple on that face.

This strict hierarchy of interface types raises another complexity unique to the multi-material case. Snaps may cause material interfaces to degenerate such that they violate the hierarchy of interface types. For example, consider a face with a triple-point. If the cuts on two adjacent edges snap to the same lattice vertex, the triple-point is now representing only a two-material interface, with a degenerate material region along the remaining line segment. To fix this degeneracy, the triple-point joins the two associated cuts at their warp destination, as in Figure 4.7. A triple-point snap may also cause cuts to become degenerate, and a quadruple-point snap may cause cuts and triples to become degenerate. The number of these cases is quite limited, and each one is tested and corrected in a way that ensures a consistent hierarchy of features and a consistent mesh.

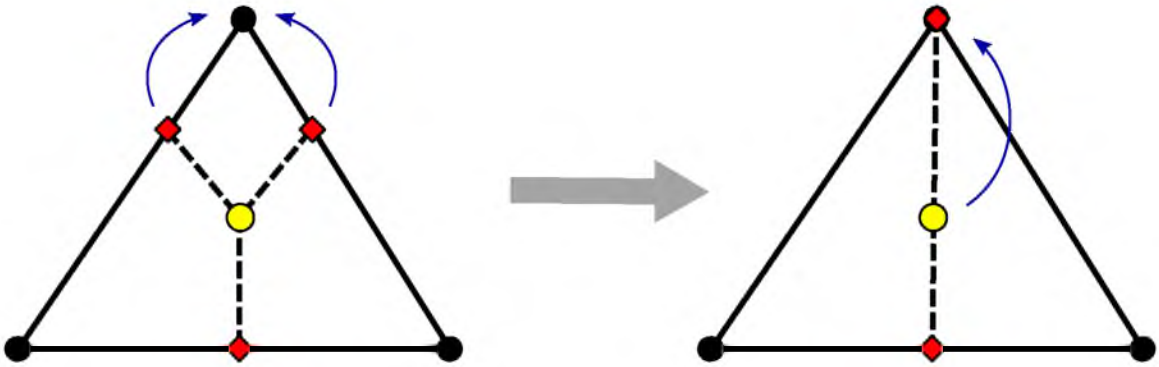


Figure 4.7. Degenerate triples or quadruples are removed by subsequent snaps.

4.8 Generalized Stencils

After snapping and warping, the polyhedra from the topological cases described in Section 4.6 may have collapsed into intermediate topologies. Each such topology demands not only a valid tessellation stencil, but one that does not permit degenerate tetrahedra when interface points are in nonviolating configurations. Moreover, each such stencil must be consistent both within the lattice tetrahedron, as well as across lattice faces (Figure 4.8). One of the contributions of this chapter is presenting a single *generalized stencil* that can be used as a master stencil for all achievable topologies. Not only does this keep the problem of stenciling local (avoiding inconsistency issues), but it also removes the requirement of implementing and storing a large stencil table that is prone to construction and transcription errors [45].

The generalized stencil is constructed from the most complicated topological case, the six-cut case. An edge is formed between every pair of points that could be snapped, ensuring the snapping procedure of Section 4.7 always simplifies the topology in a manner equivalent to a series of edge collapses. On each lattice face, edges star out from the triple-point to every edge-cut and vertex on the boundary of the face. Similarly, on the interior of the lattice tetrahedron, edges star out from the quadruple-point to every triple-point, edge-cut, and vertex on the boundary of the lattice tetrahedron (Figure 4.9). In the regular case, this is equivalent to barycentric subdivision of the tetrahedron. This construction tessellates the lattice tetrahedron into 24 stencil tetrahedra, each composed of a single vertex, cut, triple, and quadruple.

For every lattice tetrahedron that does not have this full complexity of material interfaces (six cuts), we choose vertices, cuts or triples, to have *virtual material boundaries*, as if they

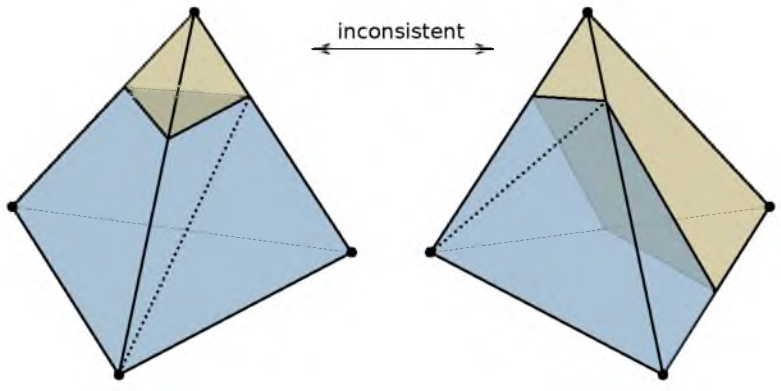


Figure 4.8. Stencils for lattice tetrahedra must be consistent across faces. In this example, the blue quad patch shared between the tetrahedra is tessellated in two different ways.

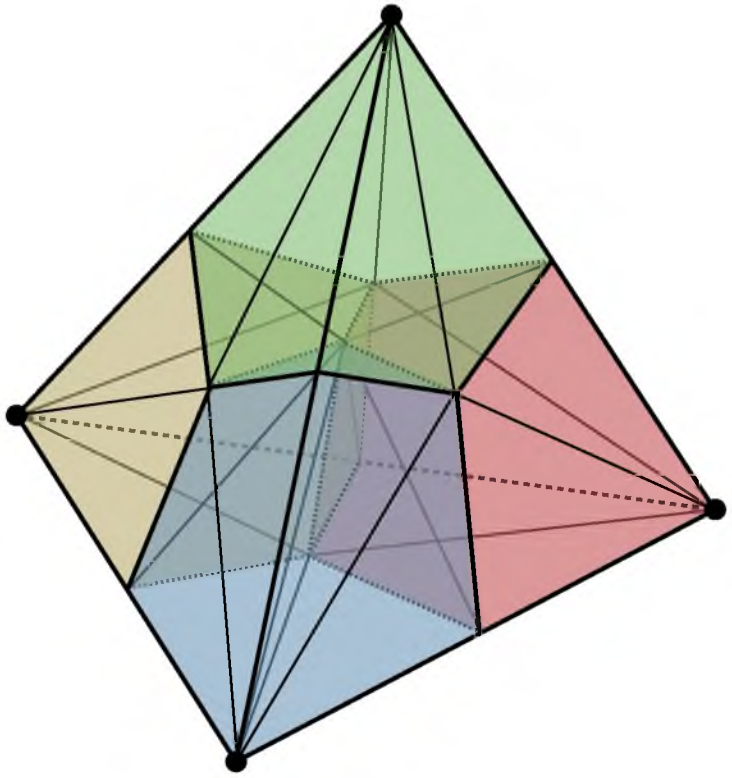


Figure 4.9. The generalized stencil is constructed from the 6-cut case. Edges connect each interface point to its associated lower order features.

had already snapped. This allows us a consistent way to tetrahedralize the polyhedra in the multimaterial stencils without worrying about inconsistencies or tangles across faces between stencils, keeping the stenciling operation local.

The procedure to generalize a lattice tetrahedron to the six-cut case proceeds as follows. First, virtual edge-cuts are created for any edge missing a cut. The remaining virtual points all cascade into place from the location of these virtual cuts. For a face that has no triple-point, we label one of the three cuts as a virtual triple-point. To maintain valid topologies, we always choose a cut that lies on an edge that already contains a virtual cut. If there is no virtual cut, a predetermined cut is chosen. Finally, if a quadruple-point is not present, we must choose a triple-point location to represent the virtual quadruple-point. We pick a triple-point using the same method a triple uses to pick a cutpoint. If there are multiple options, we choose the point that is collocated with the most other points, virtual or real.

Note that these rules for generalizing lattice tetrahedra operate once; they are merely the mechanism for generating a consistent set of stencils. These generalizations can even be computed offline, for all stencil cases, and then chosen from a lookup table at run-

time. These rules for choosing arbitrary, but consistent, transitions from the six-cut case to all of the others produce different (but valid and good quality) tetrahedralizations of the similar cut patterns, depending on their orientation on the BCC lattice. This mechanism for ensuring consistency is a multimaterial alternative to the *parity* scheme used in the two-material case [62]. This procedure as described is merely one solution within a space of valid possibilities.

4.8.1 Grading

Like other meshing algorithms built upon the BCC lattice, we take advantage of its structure to achieve graded meshes through the use of an octree. Any cell that contains at least one edge-cut is created as leaf in this tree. We utilize the same stencil as found in Isosurface Stuffing, where bisected and quadrisected BCC tetrahedra are used to span neighboring octree cells that differ in height by 1 level. Because these graded tetrahedra only appear in regions of homogeneity, the introduction of multiple materials has no affect on their structure.

4.9 Algorithm

The full lattice cleaving algorithm utilizes the rules developed in this section and proceeds as follows: Using an octree structure to reduce storage and allow grading, we first sample and label each BCC lattice vertex. Alternatively, a search strategy can be employed at this point to reduce storage and time searching for interfaces. If a tie occurs, we add an epsilon adjustment to a random material, creating cuts one or more incident edges. Next, cuts, triples, and quadruples are computed for each lattice tetrahedron that has multiple unique material labels on its vertices. The octree stores leaves only for the cells containing cuts, and it is balanced such that neighboring cells never differ by more than a height of one.

Any lattice tetrahedron that is not the six-cut case is then generalized to be so, by labeling the locations of virtual cuts, triples, and quadruples. This operation completes in a single pass over the lattice tetrahedra, either directly or through the use of a precomputed lookup table. Once all lattice tetrahedra have been generalized, three phases of snapping and warping begin.

In the first phase, all violated lattice vertices are identified and visited exactly once. Any violating interfaces on incident edges, faces, or tetrahedra are snapped to the vertex, and the vertex warps to the center of mass of the interfaces, distributing any round-off equally. All adjacent nonviolating interface points are projected to remain on their respective simplices.

If an oracle is available, the new interface points should be queried directly. All degeneracies are fixed with additional snaps, as described in Section 4.7. After completion of this phase, all lattice vertices will be free from violations.

In the second phase, all violated lattice edges are identified and visited exactly once. If one or more triples or quadruples violates an edge, we snap them to the cut on that edge, wherever it may be. Sometimes a triple-point violates an edge that no longer contains a cut, because it has already snapped to a lattice vertex. In such cases, the triple-point would snap to that lattice vertex as well. These snaps are designed so that a lattice edge may contain singular points of transitions or be a single material across its entirety; we do not allow material interfaces to lie on the half-edge. An alternative approach would be to project these triple-points to the nearest location on the edge, and split the edge with a new cut, tessellating the output stencil tetrahedra incident to that edge. This would provide increased fidelity but also increased element count.

In the final phase, we address the problem of quadruple-points that are too close to lattice faces. Using the face violation condition, we snap any such quadruple-point to the triple-point on the face that was violated. Similar to the second phase, sometimes a quadruple-point violates a lattice face that no longer contains a triple-point. It may have snapped to an edge-cut, or to a vertex. The quadruple-point always follows the triple-point, maintaining the hierarchy of features on each edge and face. Again, a splitting procedure could be utilized to retain higher fidelity.

Finally, we output all stencil tetrahedra that contain four unique vertices, skipping over any that were collapsed during the warping and snapping process. Octree cells at higher levels are also filled with appropriate tetrahedra for grading, though there is no need to delay this step until the end.

The complexity of this algorithm is worst case $O(n)$ where n is the number of voxels in the input image. However, in practice, it is rare that a set of interface surfaces would fill the entire volume. We find the complexity is most often sublinear. Additionally, while we implemented this algorithm as an in-core solution, requiring memory for the whole volume and mesh throughout the algorithm, this need not be the case. All of the operations act locally on sets of adjacent lattice cells. Therefore a streaming solution, or moving window approach, could also be employed to process lattice cleaving. The benefits of such an approach would include reduced memory footprint, and possibly better cache performance.

4.10 Bounded Dihedral Angles

The algorithm described in Section 4.9 is designed to ensure that both dihedral angles are bounded and geometric distortion of input surfaces is controlled. In what follows, we prove these properties hold true.

The violation rules defined in Section 4.6 disallow vertex positions that could lead to undesirable tetrahedra. These proofs rely on these carefully designed rules for vertex placement, a particular set of properties in the generalized stencil set, and their interaction with the background lattice.

There are multiple ways to classify types of bad tetrahedron [9, 24]. One useful partitioning groups such tetrahedra into two sets: tetrahedra whose vertices are nearly collinear, and tetrahedra whose vertices are nearly coplanar. This classification includes not only tetrahedra with bad dihedral angles, but also tetrahedra with bad solid angles (i.e., the spire).

It is also useful to classify the types of bad triangular faces that can occur on these undesirable tetrahedra, namely, *daggers* and *blades*. These triangles have vertices that are nearly collinear. While a tetrahedron may still be badly shaped without their presence, (e.g., slivers), a tetrahedron that contains poor-quality triangles will itself also be of poor quality.

The rules comprising the proposed algorithm make it impossible for output tetrahedra to become badly shaped (and consequently, they have bounded dihedral angles). First, we show that background lattice tetrahedra stay of good quality. This property induces constraints on the polyhedra of the output stencils. Finally, we show that these constraints, combined with the violation conditions for warping and snapping, always lead to tetrahedra with bounded dihedral angles. Unlike the computational proof of Labelle and Shewchuk, which relies on interval arithmetic and a numerical search, this approach does not give a specific angle bound. The multimaterial problem introduces enough additional degrees of freedom to make a similar computational approach currently intractable. This direct proof does, however, provide insights into why this algorithm is successful at achieving bounded dihedral angles and gives a foundation for modifications and extensions. We begin by introducing several definitions.

Definition 1 *A* dihedral angle θ *is the angle between two planes.*

A tetrahedron contains 6 internal dihedral angles. The dihedral angle between triangular faces can be expressed as a function $\Phi : V^2 \mapsto \Re$ of the face unit normals $\hat{n}_1$ and $\hat{n}_2$:

$$\Phi = arccos\,(\hat{n}_1 \cdot \hat{n}_2) \tag{4.1}$$

Definition 2 *The* aspect ratio, $\mathrm{ar}_f = \frac{h}{l}$, *for a triangular face,* f, *where* h *is the height of the shortest altitude and* l *is the length of the longest edge.*

Definition 3 *The* aspect ratio, $\mathrm{ar}_t = \frac{h}{l}$, *for a tetrahedron,* t, *where* h *the height of the shortest altitude and* l *is the length of the longest edge.*

For triangles, the aspect ratio goes to zero as the vertices approach collinearity. For tetrahedra, aspect ratio is a measure for how close the vertices of a tetrahedron are to being either collinear or coplanar. It turns out that when tetrahedra degenerate in these ways, it must be the case that there are either dihedral angles of 0°, 180°, or both. All unwarped BCC lattice tetrahedra have aspect ratios $\mathrm{ar_t} = 0.866025$, and dihedral angles of 60° and 90°. We next define the notion of ϵ-good.

Definition 4 *For* $\epsilon > 0$, *let* $\theta_{\min}$ *and* $\theta_{\max}$ *be the minimum and maximum dihedral angles for all possible tetrahedra with* $\mathrm{ar}_t > \epsilon$. *A dihedral angle,* θ, *is called* ϵ-good *if and only if* $\theta_{\min} \leq \theta \leq \theta_{\max}$. *Similarly, let* $\phi_{\min}$ *and* $\phi_{\max}$ *be the minimum and maximum interior angles, respectively, of all triangles with* $\mathrm{ar}_f > \epsilon$. *We call a planar angle,* ϕ, ϵ-good *if and only if* $\phi_{\min} \leq \phi \leq \phi_{\max}$.

Lemma 1 *For a triangle,* t, *with minimum and maximum interior angles* $\phi_{\min}$ *and* $\phi_{\max}$, $\mathrm{ar}_f > 0$ *iff there exists* $\kappa > 0$ *such that* $\kappa < \phi_{\min}$ *and* $\phi_{\max} < 180° - \kappa$.

Proof : Let t be a triangle composed of vertices v_1, v_2, and v_3. Let L be the length of the longest edge, joining v_2 and v_3, and H be the height of the shortest altitude, incident to v_1. v_1 can be moved along the line parallel to edge v_2v_3, without changing the height of this altitude (Figure 4.10). If $H > 0$, a planar angle ϕ can only approach 0° or 180° as it moves infinitely in either direction. However, as v_1 moves in either direction, either edge v_1v_2 or v_1v_3 lengthens. Eventually, this length will become longer than L, and the aspect ratio will change. Therefore, the movement of v_1 is bounded in order to keep ar_f fixed. The worst minimum angle possible becomes $\phi_{\min} = \arcsin(\mathrm{ar_f})$, and the maximum $\phi_{\max} = 180° - \arcsin(\mathrm{ar_f})$.

Lemma 2 *For a tetrahedron,* t, *with minimum and maximum dihedral angles* $\theta_{\min}$ *and* $\theta_{\max}$, $\mathrm{ar}_t > 0$ *iff there exists* $\kappa > 0$ *such that* $\kappa < \theta_{\min}$ *and* $\theta_{\max} < 180 - \kappa$.

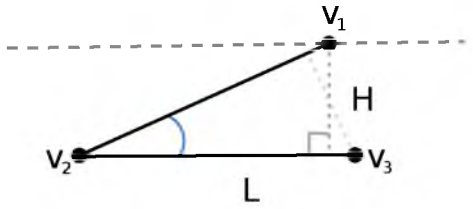


Figure 4.10. The space of triangles with aspect ratio H/L. v_1 is restricted to move along the axis parallel to edge v_2v_3 such that L and H do not change. This 1D space is bounded on both sides.

Proof : The dihedral angle, θ, between two incident triangular faces may be computed using the 3D face normals, as in Equation 4.1. There is an equivalent planar angle, ϕ, formed by projecting these faces along the axis coinciding with their shared edge (Figure 4.11).

The lengths of the two line segments that form this angle in 2D are equivalent to the lengths of the two triangle altitudes. The distance of the projection of each vertex not incident to the shared edge, to the opposite face becomes the height of an altitude in 2D projective space. This gives an equivalent equation for both the planar and dihedral angle associated with these vertices:

$$\theta = \arcsin\left(\frac{h_t}{h_f}\right) \tag{4.2}$$

where h_t is the height of the tetrahedron altitude in 3D (and equivalently the height of the triangle altitude in 2D), and h_f is the height of the altitude of the incident face (and equivalently the length of the incidentedge in 2D).

Because the altitudes are orthogonal to the shared edge, by the Triangle Inequality, the lengths of the two altitudes are bounded by the edge lengths of the two triangular faces. Let L be the longest edge of the two faces and H the shorter of the two 3D altitudes between them. It must be the case that $\arcsin\left(\frac{H}{L}\right) <= \arcsin\left(\frac{h_i}{l_i}\right)$ for all i. Because this relationship holds for each pair of faces of a tetrahedron, t, it also must be the case that $\arcsin\left(\frac{H_{max}}{L_{min}}\right) <= \arcsin\left(\frac{h_i}{l_i}\right)$ for all i, or:
$arcsin(\mathrm{ar}_t) <= \theta <= 180 - arcsin(\mathrm{ar}_t)$ for all θ in t.

Figure 4.11. For every dihedral angle, there is an equivalent angle in 2D. (left) 3D dihedral angle (right) projected to 2D with altitude.

Lemma 3 *There exists a set of violation parameters α_{short} and α_{long} for which all BCC lattice tetrahedra maintain ϵ-good dihedral angles after warping as described in Section 4.7.*

Proof : Let t be a BCC lattice tetrahedron and r_α be the radius of a ball around each vertex. Each ball contains the possible set of points to which its vertex may warp, given the violation parameters α_{short} and α_{long} (the violation parameters for short and long edges, respectively). If $r_\alpha = 0$, no warping takes place, and t has aspect ratio $\text{ar}_t = 0.866025$. Because the worst dihedral angle of a tetrahedron can be defined as a continuous function of vertex positions, by the *intermediate value theorem*, there must exist an r_α for which $\text{ar}_t = \epsilon > 0$. Thus, by Definition 4, there must exist an α_{short} and α_{long} for which the lattice tetrahedra maintain ϵ-good dihedral angles after warping.

Definition 5 *Let p be a polyhedron subdivided into a set of polyhedra S. A polyhedral face f from the set S is considered* external *if it is incident to ∂p.*

Lemma 4 *All stencil polyhedra with nonviolating vertices maintain ϵ-good dihedral angles around edges incident to at least one external face.*

Proof : For every dihedral angle of a stencil polyhedron that spans an edge incident to an external face, either one face or both faces are external. If both faces are external, then the dihedral angle equals that of the enclosing background polyhedron. By Lemma 3, we know this is an ϵ-good dihedral angle. If one face is internal and the other external, there is at least one vertex, v_i, on the internal face that is not incident to the external face. The only way for the dihedral angle to lose the ϵ-good property is by moving v_i arbitrarily close to the external face, its edges, or its vertices. In each case, a violation condition from Section 4.6 would be triggered, making these impossible.

Lemma 5 *All output tetrahedra span at least two stencil polyhedron faces that meet at ϵ-good dihedral angles.*

Proof : In general, any two faces may either both be external, one external and one internal, or both internal. If both are external, by Lemma 4 we know that the dihedral angle between these faces will remain ϵ-good. If one is external and the other internal, by the violation conditions of Section 4.6 we know these faces are ϵ-good. The case of two internal faces is prevented through stencil selection. There are five regular topological stencil cases before snaps. Snapping can only simplify polyhedra, never creating additional internal faces. Thus, if no tetrahedra span two internal faces of the polyhedra in the regular

topological cases, it is also the case that no tetrahedra span two internal faces after edge collapses. Among the regular topological cases, only the five-cut and six-cut cases have the potential for multiple internal tetrahedron faces. The generalized stencil is designed specifically to avoid any tetrahedra spanning the faces that are not guarded by violation conditions.

Lemma 6 *All stencil triangles maintain ϵ-good planar angles after snapping and warping, as described in Section 4.7.*

Proof : All output stencils are composed from only four types of vertices: (lattice) vertices, cuts, triples, and quadruples, abbreviated v,c,t, and q, respectively. The vertices of a stencil triangle can exist in three ways. Either all three vertices lie on the same lattice face, two vertices lie on the same lattice face, or all three lie on unique lattice faces.

If all three vertices lie on the same lattice face, the violation conditions for cuts and triples guard against such an aspect ratio. There are only three sets of points that can become collinear, and the proposed stencils specifically preclude triangles spanning them: vcv, vtc, and ctc.

If two vertices lie on the same lattice face, the triangle's aspect ratio, ar_f, can only fall below ϵ if the third vertex is in violation of the face containing the other two. This includes the third vertex violating an edge of the lattice face containing the two vertices.

Finally, if all three vertices lie on unique lattice faces, the triangle's aspect ratio, ar_f, can only fall below ϵ if all three vertices are violating the vertex incident to all three lattice faces.

Lemma 7 *All output tetrahedron have ϵ-good dihedral angles.*

Proof : Let t be an output tetrahedron. By Lemma 5, t has at least two faces joined at an ϵ-good dihedral angle along an edge e. By Lemma 6 the triangles incident to edge e are ϵ-good. The existence of one good dihedral angle with two good faces incident to it implies t must have ϵ-good dihedral angles everywhere.

Theorem 1 *There exists a dihedral angle, $\theta^* > 0°$, such that the dihedral angles of all tetrahedra are bounded from below by θ^* and above by $180° - \theta^*$.*

Proof : By Lemmas 7 and 2.

This proof shows that the meshes from the lattice-cleaving algorithm never degenerate, in fact Lemma 6 also ensures we produce no bad solid angles (e.g., "spires"), despite them

lacking bad dihedral angles. Moreover, in practice, with proper choice of α, this bound, θ^*, is significant, and empirical results in Section 4.11 corroborate this fact.

4.10.1 Geometric Fidelity

We next make a statement about the quality of the surface approximation. Let Σ be the *interface surface*, the complex of smooth surface patches where two materials meet, as well as the associated curves where three materials are coincident, and the points where four meet. Σ is a CW-complex, and geometrically behaves as a piecewise-smooth complex [26]. It also has a well-defined medial axis M_Σ that we define as the closure of the set of points in $\mathbb{R}^3$ that have at least two closest points in Σ. Each point in M_Σ is the center of a ball that meets Σ only tangentially. Using the medial axis, we can quantify of the scale of features at each point $p \in \Sigma$. In particular, we define local feature size, $\text{lfs}\colon \Sigma \to \mathbb{R}$, as the distance from each surface point to the medial axis. Local feature size is well studied in smooth surfaces [3]. In the mutimaterial setting setting, local feature size approaches zero near triple junctions, which meet nonsmoothly. Consequently, we define the set of *h-regular points*, $\Sigma_h = \{p \in \Sigma \mid \text{lfs}(p) > h\}$ and restrict our claims to these.

Given a tetrahedron c in the mesh, we make a claim regarding its vertices. Let $\Sigma|_c$ be the *restriction* of Σ to c, defined as $\Sigma \cap c$. We define an *h-regular tetrahedron* c as one where the set of $p \in \Sigma|_c$ is regular.

Lemma 8 *Given an h-regular tetrahedron c constructed from BCC lattice edges which are no longer than h, any vertex v of c that is labeled as having two materials (surface vertex) lies on* Σ.

Proof : Because v has two materials, it is the byproduct of a warp and snap to bring it to the Σ. Prior to this, v underwent a sequence of operations that depended on the cuts of edges incident to v. As long as there was only one such cut, v only warped once, and directly to the surface as computed by the indicator function oracle. We prove that because c is h-regular, this is always the case. Assume, for sake of contradiction, that there were multiple cuts, of different materials, on edges incident to v (if the materials were identical, we preferentially pick the closest cut to move to). Without loss of generality, assume there are two cuts of material type AB and BC, we call x and y. Both x and y lie in the violating zone, on an edge incident to v. They are no further apart than $2h\alpha_{\text{short}} < h$. This indicates that the two surface patches defining these cuts are no further apart than h as well.

We show there must be a medial axis point within distance h of $\Sigma|c$. Consider the medial

axis for any single connected material region. By definition it is a deformation retract of this region, and in addition, it touches any point where three materials meet. Thus, within a single region, the medial axis is a single connected component which connects all triple-points. If we walk along the line segment joining x and y, we must therefore cross the medial axis, because otherwise it would violate the above property. As a result, there is a medial axis point at this crossing. This medial axis point must be within distance h of v. However, because v lies on $\Sigma|_c$, this violates the fact that c is h-regular, leading to a contradiction.

In generalizing this case to when more than two cuts are adjacent v, the same logic holds for any pair of them, which is sufficient to create the same contradiction above.

All h-regular tetrahedron are well-behaved; in general, they act just like the tetrahedra in the isosurface stuffing algorithm [62]. Most importantly, when they do mesh a piece of the surface, they geometrically approximate the surface. As with any pointwise probe, it is impossible to guarantee that there are no tiny features missed on account of the mesh resolution being too coarse. However, we can guarantee that for h-regular tetrahedra containing surface patches of Σ, every point on an interface triangle representing this patch lies close to Σ. This "one-sided" notion of distances mirrors Theorem 2 of [62] (only the distance of every mesh vertex to Σ is bounded). When the mesh is of fine enough resolution and each surface patch of Σ sufficiently smooth, the distance bound for h-regular tetrahedra becomes two-sided—the claim follows for distances from Σ to the mesh.

4.11 Results

Our implementation of lattice cleaving is extremely fast, requires virtually no user interaction, and achieves bounded dihedral angles. These bounds depend on choices of the violation parameters α_{long} and α_{short}. In this section, we discuss the importance of these choices, demonstrate the lattice cleaving algorithm on various datasets, and discuss other aspects of the algorithm.

4.11.1 Parameter Choice

The α_{long} and α_{short} parameters determine the distances along an edge, beyond which an edge-cut is considered "safe", or nonviolating. They decide the trade-off between snapping/warping and stencil cleaving. A user who is primarily interested in visualization might choose to turn off violations completely by using $\alpha_{\text{long}} = 0$ and $\alpha_{\text{short}} = 0$. This will give the most accurate geometric representation with possibly degenerate elements. As the alpha values increase, the severity of the worst possible dihedral angles decreases, up until a point.

With sufficiently large alpha values, lattice tetrahedra may become flat or even inverted. If a user is interested in only one side of the interface, flat elements can be tolerated by stripping them from the surface.

Because our meshes represent volumes on both sides of each interface surface, the most appropriate parameters would seem to be those used by Labelle and Shewchuk [62] in the *double-sided* surface case. However, our multimaterial violation conditions use slightly more conservative bounds to take into account the dihedral angles near triples and quadruples. We picked the parameters $\alpha_{\text{short}} = 0.357$ and $\alpha_{\text{long}} = 0.203$ and found they achieved a worse case minimum angle of 2.76° and worst case maximum angle of 175.426°. In practice, after simulating many hundreds of times steps of several dozen fluid simulations, corresponding to hundreds of millions of tetrahedra, we see much better angles for the vast majority of meshes.

4.11.2 Aliasing

A fundamental aspect of the lattice cleaving algorithm is that it operates at a fixed resolution. Any feature that falls below grid width will not be captured in the output mesh. For two-material interfaces, this simply results in the smoothing of a surface, or removal of small topological features. For three-material interfaces, the behavior becomes more interesting. As the feature size around a three-material interface always goes to zero, there will always be some degree of approximation. But beyond that, the interaction of the background lattice faces with this approximation can lead to regular patterns of topological aliasing.

Figure 4.12 illustrates a scenario that can arise in 2D. In the scenario, two material interfaces come together at a sharp corner. Because the BCC lattice contain vertices from two sets of regular grids, (primal and dual), neighboring lattice faces alternate having either two or three unique material labels on their vertices. This leads to a saw-tooth-like pattern of spikes and pillars that form topologically distinct regions.

These aliasing artifacts are sometimes taken care of through the snapping and warping procedure, because the sharp spikes that form are often in a violating condition anyway. However, because there is a space of cases which are not handled by snapping, explicit solutions for this problem are needed. A range of possible solutions exist, such as smoothing, tightening [111], or morphological operations such as dilation and erosion. If the input data cannot be smoothed, one could also design discrete local operators that intelligently change material labels on lattice vertices to avoid aliasing.

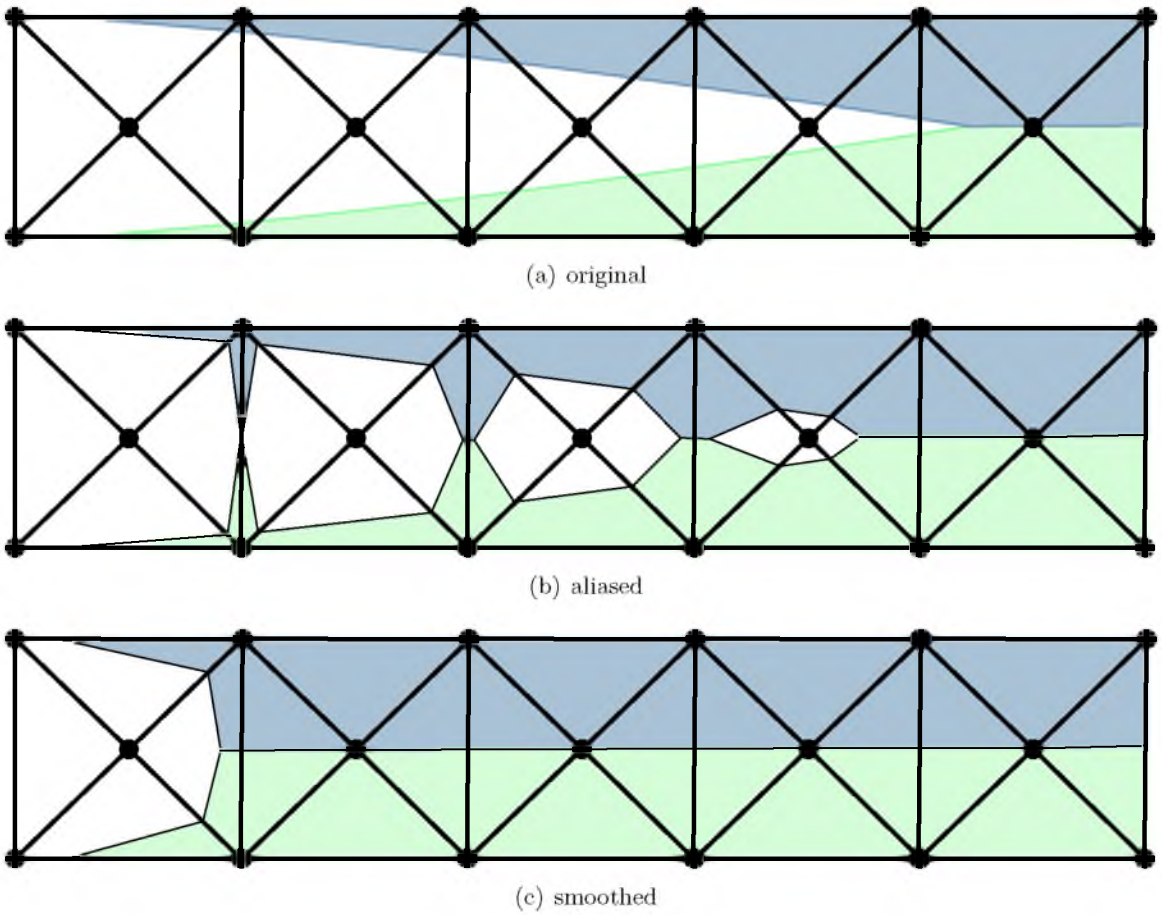


Figure 4.12. Sharp features can interact poorly with the background lattice. (a) Three materials meet in a sharp corner feature. (b) Using material labels at each lattice vertex results in aliasing artifacts. (c) preprocessing (e.g., smoothing) indicator functions eliminates this problem.

4.11.3 Examples

To illustrate some of the datasets for which lattice cleaving can be used, we provide several examples. Run times for these datasets were calculated on a machine with an Intel Core i7 3.2 GHz CPU and 12GB of RAM.

Figure 4.13 shows a mesh generated from a segmented MRI scan of a human head. The algorithm completed in about 100 seconds and produced a mesh with roughly five million elements, all with dihedral angles between 4.33° and 157.98°. Figure 4.14 shows a mesh generated from a similar scan of a human torso. The algorithm completed in under a minute and produced a mesh with roughly 12 million elements, all with dihedral angles between 5.11° and 159.91°.

It is often necessary to visualize multimaterial tetrahedral meshes before any simulation work is conducted. This spans from the need to qualitatively verify results are accurate, to

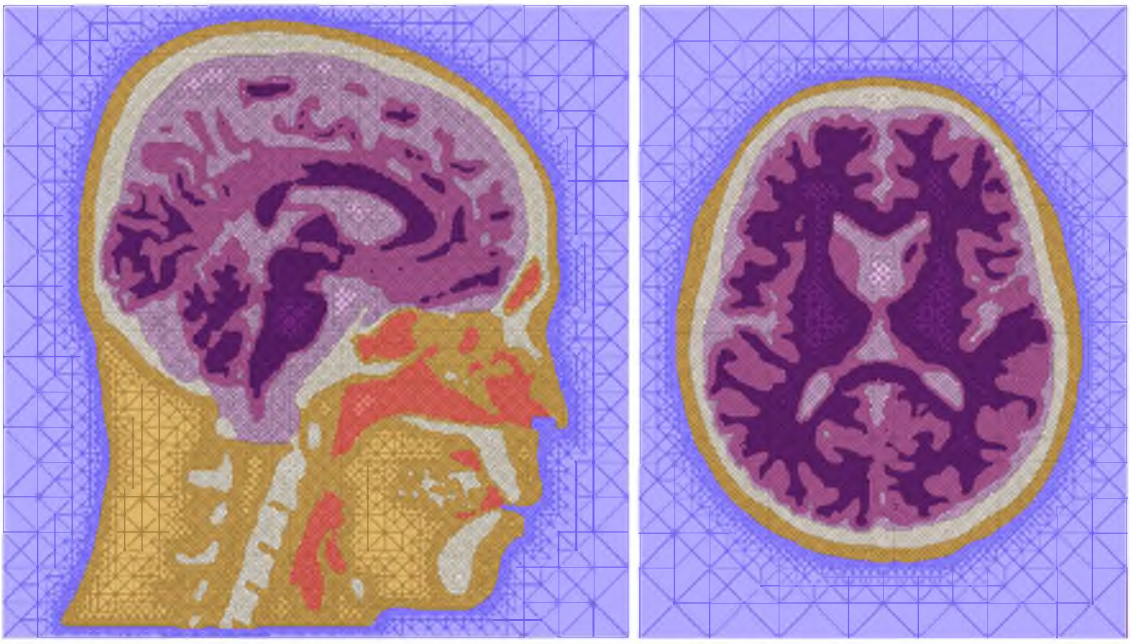


Figure 4.13. Meshes generated from MRI scan of a human head. Resolution: $264 \times 264 \times 264$. Dihedral angles: $[4.33° - 157.98°]$. ≈ 5.3 million elements.

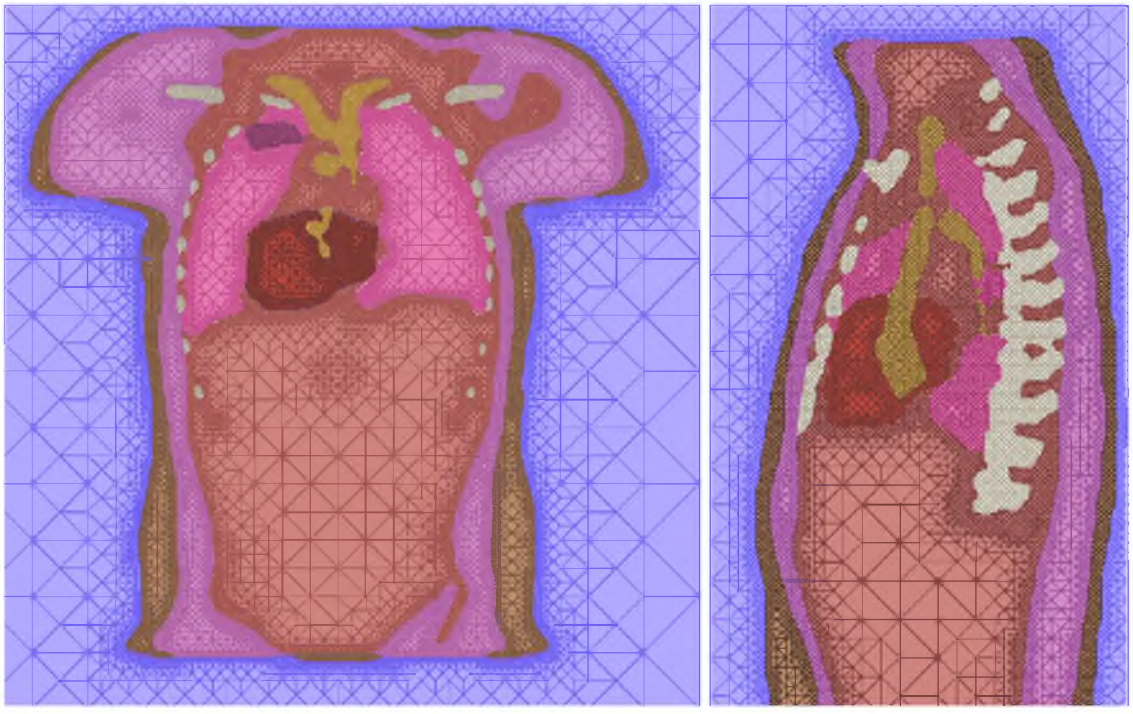


Figure 4.14. Meshes generated from MRI scan of a human torso. Resolution: $208 \times 96 \times 208$. Dihedral angles: $[5.11° - 159.91°]$. ≈ 12.6 million elements

spotting unexpected features that might influence solutions. Figure 4.15 shows a visualization generated from a segmented frog MRI. The input volume was $260 \times 245 \times 150$ in size, and took just over a minute to mesh. The surface meshes can be extracted from the lattice cleaving algorithm as a postprocess or generated alongside the tetrahedral mesh using the same stencil set.

This work also applies to multiphase fluid simulation and animation. To demonstrate this, we utilize the lattice cleaving algorithm in the core of a multiphase viscous fluid simulation. Figure 4.16 shows a rendering and cutaway view of the underlying mesh used for physics. This simulation used a 64^3 background lattice (primal vertices), required 8 seconds to mesh, and produces, on average, 1.2 million tetrahedra. Figure 4.17 shows a histogram of the dihedral angles generated from 350 simulation frames. The majority of elements are of excellent quality, with small tails near the expected bounds. Counts for angles belonging to unwarped background lattice tetrahedra are scaled down.

4.11.4 Algorithm Comparison

We evaluate the proposed meshing algorithm against two other packages commonly used in biomedical meshing: BioMesh3D v1.0 [11, 76] and CGAL v3.9 [104]. BioMesh3D uses variational optimization [76] to sample a domain and TetGen [100] to construct a Delaunay tetrahedralization of those samples. CGAL's 3D mesh generation package contains an engine based on Delaunay refinement [34, 86, 96].

We compare these methods through their performance in three state-of-the-art simulation experiments. The first simulation is from an osseointegrated bone implant experiment, studying the effects of using direct current cathode stimulation to enhance the ability of implants to fuse into the skeletons of rabbits [55]. The second simulation is used for modeling the effects of EEG source on a human skull [37]. The third simulation used is for modeling ICD placement for defibrillation in children and adults [107].

Tables 4.1–4.3 show the results of the simulations for these three biomedical applications, using the meshes generated by the three methods. For each of these experiments, we tuned parameters of the different system to get approximately the same number of elements (within a factor of 2). We analyzed the geometric quality of the mesh (dihedral angles), the condition numbers of the resulting stiffness matrices, and the number of iterations the solver took to converge. All of these simulations use the finite element method with linear elements and they are solved using the SCIRun 4.5 [91] implementation of the minimal residual (MINRES) method [80] (a variant of the conjugate gradient method), combined with a Jacobi preconditioner.

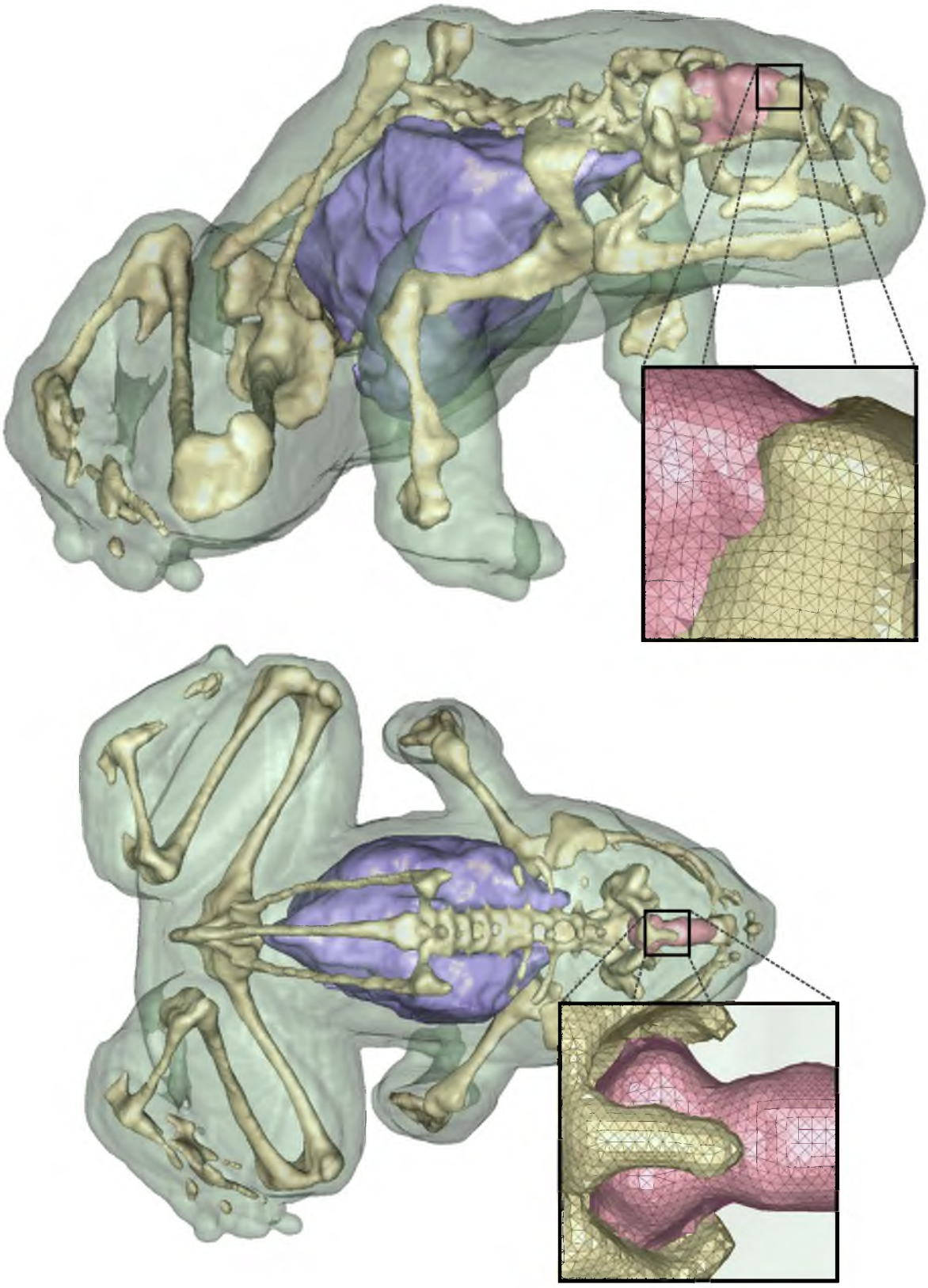


Figure 4.15. Visualization using surface meshes generated from a segmented frog MRI. Resolution: $260 \times 245 \times 150$. Dihedral angles: $[6.06^\circ - 154.28^\circ]$. ≈ 14.8 million elements.

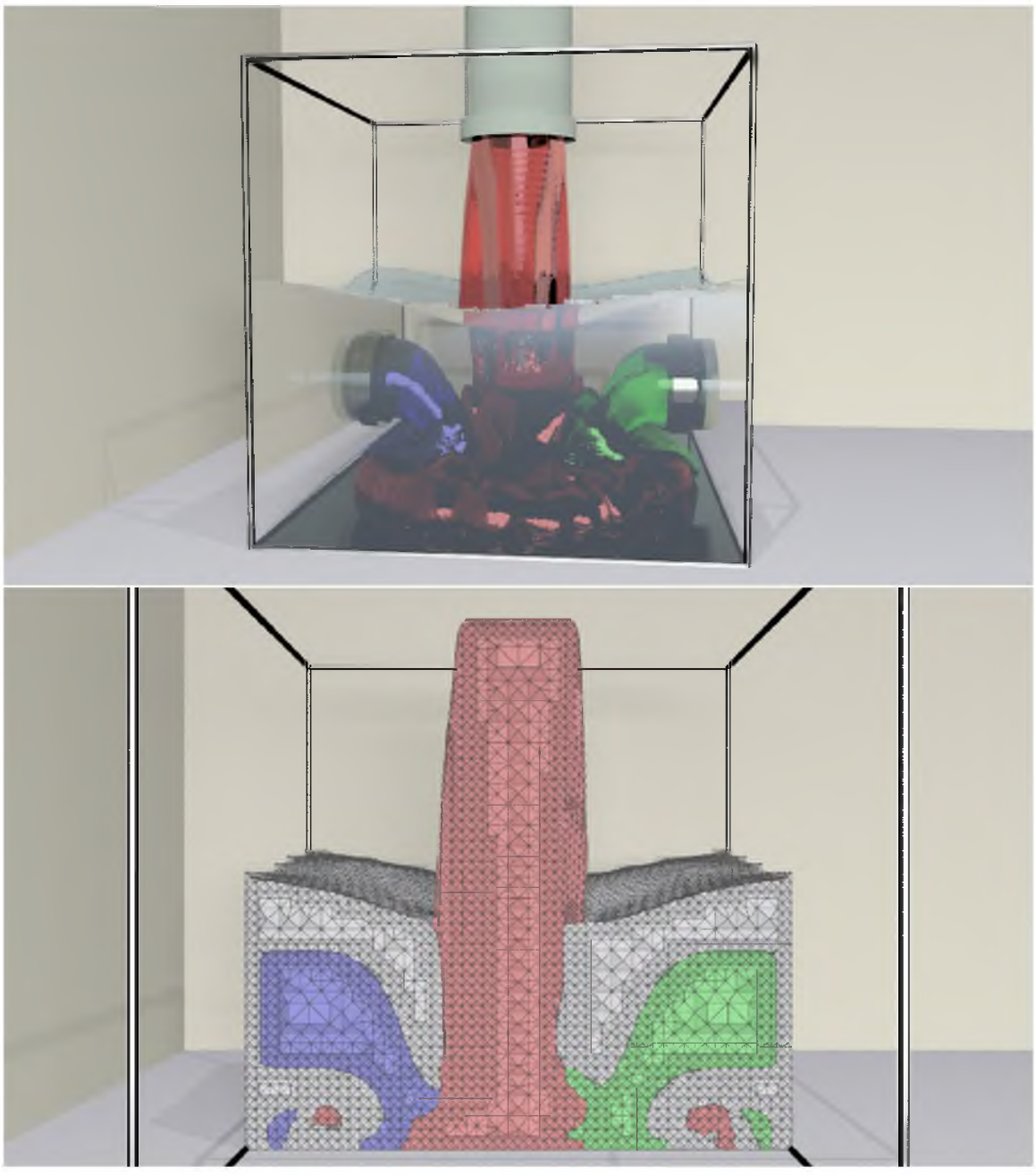


Figure 4.16. A multiphase viscous fluid simulation. Each frame uses a conforming mesh to compute fluid physics.

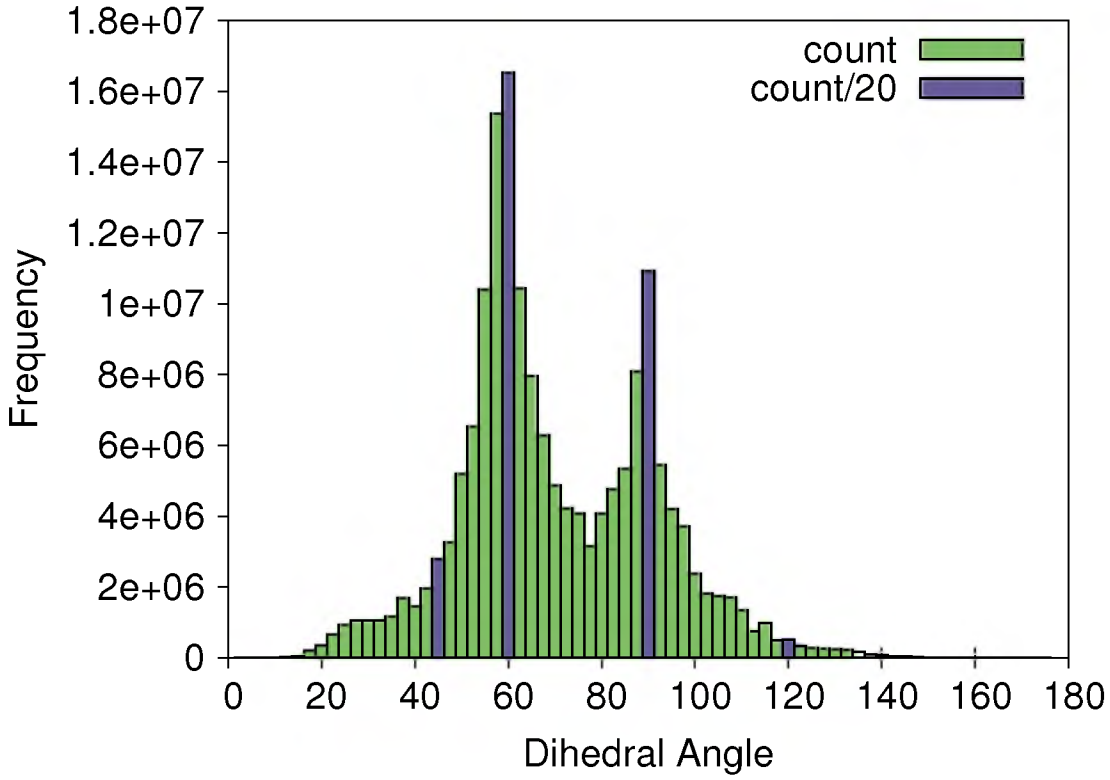


Figure 4.17. A histogram of all the angles produced through the fluid simulation. Purple bars have 20x the counts shown.

Table 4.1. Torso Simulation: 10 materials, $208 \times 96 \times 208$

Method	Elements (m)	Min Angle	Condition	Iterations
Cleaving	12.6	7.43	5.42e+06	553
BioMesh	22.1	0.00	2.94e+09	399
CGAL	14.5	0.04	2.42e+07	1008

Table 4.2. Head Simulation: 8 materials, $264 \times 264 \times 264$

Method	Elements (m)	Min Angle	Condition	Iterations
Cleaving	11.7	6.49	1.57e+18	493
BioMesh	7.9	0.18	7.49e+18	525
CGAL	11.5	0.01	1.04e+20	1026

Table 4.3. Rabbit Leg Simulation: 6 materials, $\times 520 \times 520 \times 300$

Method	Elements (m)	Min Angle	Condition	Iterations
Cleaving	5.5	7.80	1.22e+07	728
BioMesh	4.3	0.53	1.12e+07	1143
CGAL	5.6	0.03	8.94e+07	1490

We see from these simulations that the proposed meshing algorithm gives better overall bounds on dihedral angles and thereby avoids badly shaped tetrahedra. For the simulations, the proposed mesher and and BioMesh perform similarly, while both mesh techniques generally outperform CGAL, requiring roughly half the number of iterations for the numerical solvers to reach convergence.

Some of these differences are also seen qualitatively looking at renderings of these meshes, as in Figures 4.18–4.20. Here we see the rough, aliasing-type artifacts that result from the fact that CGAL operates only on discrete label maps—a reasonable explanation for the differences in simulation outcomes. Qualitatively, the proposed method and Biomesh are similar, but exhibit different strategies on grading/adaptivity. Besides the bounds on dihedral angles, the proposed method has significant advantages in reliability, robustness, and ease of use. All of the results for the proposed method were produced in one pass using one free parameter, which is the resolution of the finest mesh, and each took between 30 to

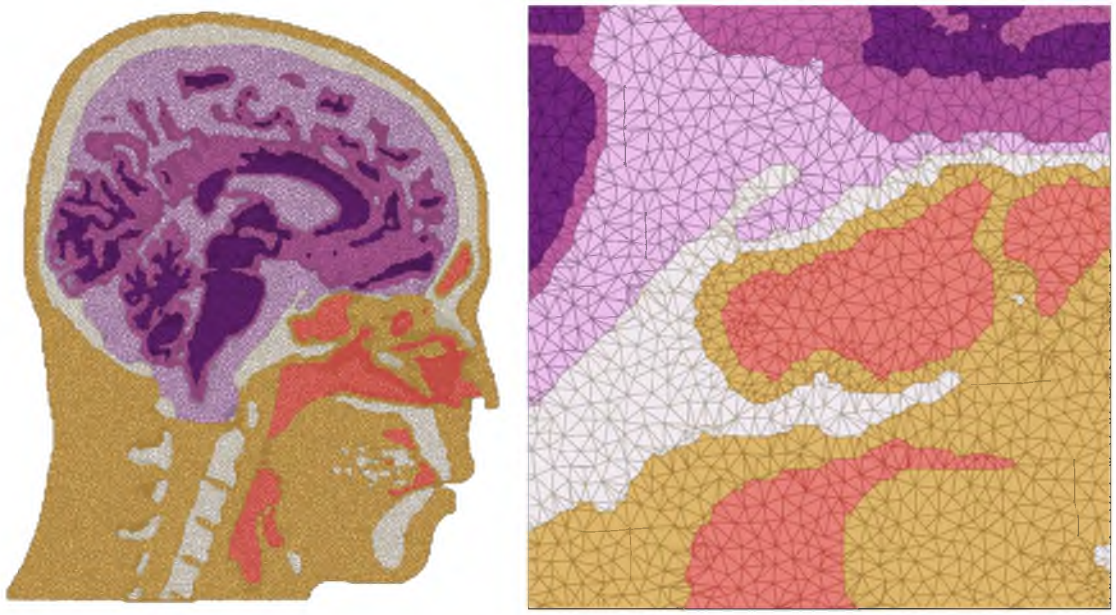


Figure 4.18. Cross-sections of the tetrahedral head meshes generated using CGAL.

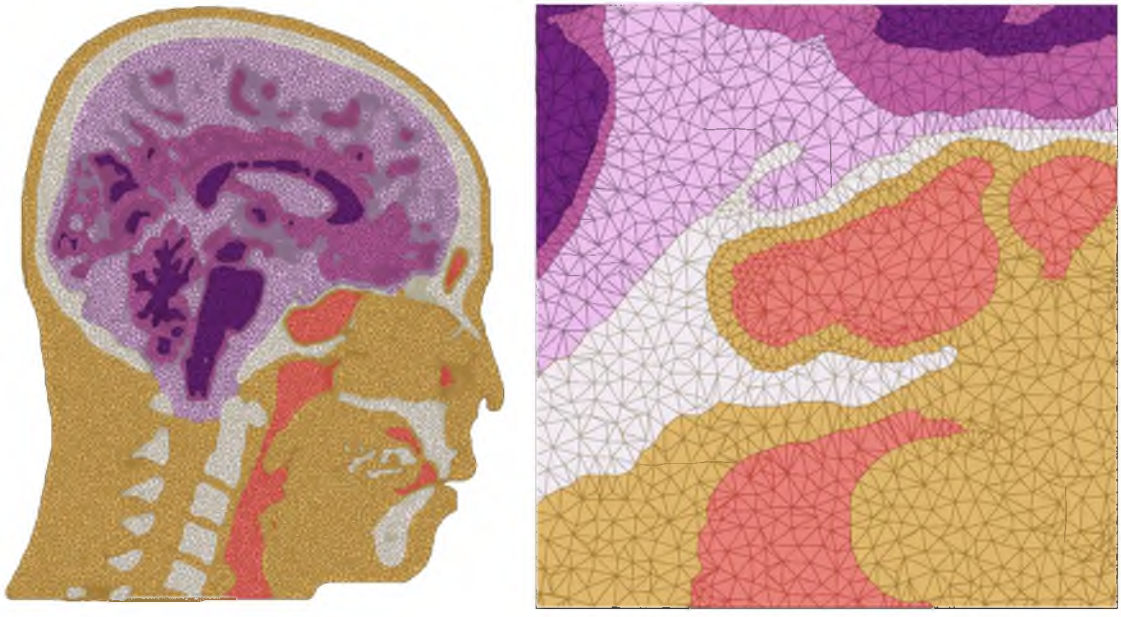


Figure 4.19. Cross-section of the tetrahedral head mesh generated using BioMesh3D.

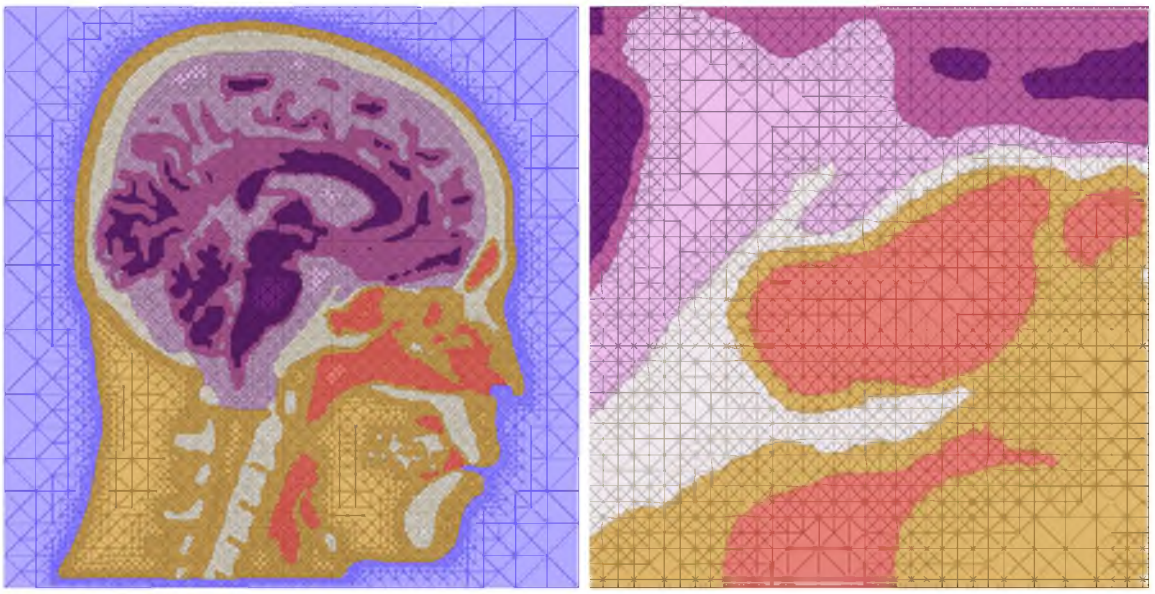


Figure 4.20. Cross-section of the tetrahedral head mesh generated using Lattice Cleaving.

300 seconds. CGAL performance was competitive, each mesh taking only a few minutes to complete. BioMesh3D performs a series of variational optimizations with a set of interacting particles, and has a variety of free parameters. The BioMesh3D results required trial and error testing for parameter tuning in order to obtain valid mesh results. Each full mesh took over 12 hours to compute.

4.12 Discussion

We have developed an extremely fast and robust conforming tetrahedral meshing algorithm for multimaterial volumetric domains. This method is guaranteed to produce meshes with bounded element quality and empirically this bound is significant, between 2.76° and 175.426°. In practice, angles tend to be much better. An open-source implementation of this method, called Cleaver [90], is now available.

The method falls under the category of stenciling algorithms, operating locally on portions of a volume. As such, it is highly parallelizable and amenable to hardware acceleration. Moreover, a single generalized stencil is used for all stencil cases, removing the need for error-prone case tables while ensuring consistent meshes.

An octree structure is used to reduce element count by providing didatic grading in homogenous regions. Future work worth exploring is achieving grading on interface surfaces, which may be sufficiently smooth as to not require finest grid resolution elements. Similarly, alternative background lattices and stencils should be examined for the purpose of achieving

anisotropic elements, which are invaluable for particular domains.

The simplifying restriction of at most one material transition per edge was instrumental in making this problem tractable. However, this restriction also places requirements on the smoothness of the input surfaces in order to avoid artifacts. Allowing for up to two transitions per edge might also relax this smoothness restriction significantly, if a suitable set of safe and compatible stencils can be constructed.

As formulated, the algorithm is oblivious to vertex ordering. The same mesh, provided to the algorithm in a different vertex/element input order, can produce slightly different output meshes. This might be an undesirable property for some needs, and could be overcome by enforcing an artifical ordering on the vertices, such as material priority or spatial ordering. Similarly, for domains that require perfectly symmetric meshes, this symmetry could be encoded into the vertex/element ordering to ensure that the algorithm operates symmetricly across the input domain.

CHAPTER 5

UNSTRUCTURED AND ADAPTIVE CLEAVING

In this chapter, we propose a new strategy for building boundary conforming tetrahedral meshes. We take an approach of explicitly decoupling the problem of conforming the mesh from all other aspects of meshing. This strategy is based on the observation that several somewhat effective methods exist for adaptive, and even anisotropic meshing, but they typically have difficulty at boundaries. We observe that in the absence of the need to conform to a boundary, many other aspects of meshing become much easier to achieve.

The proposed strategy is to first build a background mesh with the appropriate tetrahedral properties in terms of size and shape. Next, we apply a single *cleaving* step which conforms the mesh to a boundary without greatly disturbing the characteristics of elements in the initial mesh. Near where we conform, we do expect some degradation to occur; however, the effects are minimized through a carefully designed set of mesh warping and improvement operations.

In developing this new framework, we make several technical contributions including a new method for building graded, unstructured meshes as well as a generalization of the isosurface stuffing and lattice cleaving algorithms to unstructured background meshes. By starting with an input mesh satisfying constraints on the elements first, we observe that the technique for making it conform to a boundary leads to only a bounded degradation in the nature of the elements. Moreover, we never need to iterate, for example between improving element quality and recapturing the boundary. This result is a generalization of those reported by the well-known isosurface stuffing [62] and lattice cleaving algorithms [19]. Both algorithms show exactly the same behavior, albeit for a more limited class of boundaries (in the case of isosurface stuffing) and input background meshes (both cases require BCC inputs). By decoupling the problems of satisfying element and boundary constraints, we treat meshing for elements of a particular type as a preprocess, enabling a broad range of

meshing algorithms to effectively ignore boundary constraints, which can then be satisfied using the cleaving technique.

5.1 Background

The automatic construction of volumetric meshes has for decades been a relevant problem in computational science and engineering. However, despite numerous successes, it would seem the state-of-the-art is still a "mesh *du jour*" strategy, where for each new computational effort, an expert is needed to construct, either automatically or in practice semi-automatically, an appropriate domain discretization. One potential explanation for such an approach is that no one meshing algorithm can serve all needs, because no single set of constraints encompasses the necessary properties of every possible downstream application for which a mesh will be used.

Consequently, one of the most desirable properties in a meshing algorithm is the ability to construct meshes which are useful in a broad range of settings. This search for additional algorithmic flexibility is one explanation for the popularity of unstructured meshes of triangles and tetrahedra, in spite of their apparent deficiencies in potential solution quality relative to quadrilateral/hexahedral meshes [8, 35, 38, 93]. Indeed, the simple need for an end user to build *any mesh at all* often trumps more sophisticated concerns. Satisfying (automatically) a complex set of meshing constraints on element size, shape, number, type, alignment, and/or orientation while still capturing salient boundary conditions is a formidable task. Individual constraints often conflict with each other. For example, meshing with perfectly isotropic elements strictly prevents the ability to vary the size, because to shorten one edge relative to others would create some slight amount of anisotropy. Other constraints act in concert, for example, matching a global alignment field can naturally lead to anisotropic shapes of elements.

Techniques that "start with a boundary conforming step and then create elements to mesh the interior" often suffer because preserving the boundary frustrates efforts to achieve other element characteristics. Because the boundary constraint may be too rigid, there may no longer be enough degrees of freedom to build a mesh with the remaining prescribed characteristics. Both the amount of work needed to fix the few bad regions, as well as final amount of improvement in element can sometimes be unknown. For example, Delaunay-based approaches for first meshing surfaces (both smooth and piecewise-smooth) followed by volume refinement typically lead to slivers [13, 14, 25, 26, 27]. In this setting, the core challenge is that removing slivers may require iterating between improving element and

boundary quality [1, 106] or other complex techniques [24].

With the exception of lattice-based approaches, when meshing to conform to a boundary, the majority of meshing algorithms first try to capture the boundary constraint. Typically, such meshes are produced by meshing boundary features in an increasing dimensionality. For example, if the domain contains sharp features, meshes are first constructed for the 0- and 1-dimensional feature curves, followed by 2-dimensional feature surfaces, and then the embedding volume is meshed with elements. This paradigm of meshing is natural for advancing front techniques, because the boundary elements provide a seed surface from which to grow the front. We remark that an interesting conclusion, parallel to this work, in the domain of hexahedral meshing by advancing front (paving and plastering) is that relaxing the boundary constraint by delaying boundary meshing leads to improved results [102].

Meshing by sequencing through boundaries of increasing dimension is particularly popular for tetrahedral meshing in the presence of complex non-manifold boundaries (both by Delaunay and variational methods, or combinations of them [14, 17, 25, 76, 106]). However, as dimension increases, the collection of lower dimensional elements impose an increasingly complex set of constraints for the next stage of meshing. In terms of Delaunay refinement, this typically means that the placement of locations for new vertices becomes limited. For example, the insertion of points to improve elements may be blocked because they would otherwise disturb boundary features [14, 25, 26]. In terms of variational approaches, this means that the next stage of optimization becomes a more expensive constrained energy minimization [1, 17]. Alternatively, one can allow the boundary to be disturbed, but then iterate in attempt to reconform to the boundary. [106]. Nevertheless, the Delaunay-based techniques often offer provable guarantees on the ability to conform to piecewise linear [96], smooth [13, 27], and piecewise-smooth [14, 25] boundaries.

While guaranteed meshing with Delaunay-based approaches is an seminal advancement of the meshing community, where the proofs break down motivates a new approach. Often, these algorithms have only limited results on the characteristics of elements lying near the boundary. For example, a common property is a bound on the circumradius to shortest edge ratio (naturally improved by the Delaunay property). However, this ratio does not optimize for quality in terms of dihedral angles, and thus the creation of sliver elements is possible. Slivers are created when sampling a smooth surface and meshing volumetrically, because a dense enough sample on the surface creates a number of almost-planar placements of vertices. In the simplest approach, inserting the circumcenter of a sliver tetrahedra [97]

in practice destroys slivers, but in the context of Delaunay meshing, it cannot be proved that this process terminates nor holds near a boundary [61]. More sophisticated techniques for sliver exudation exist which both require expensive computational tools and can only offer limited improvement (the very worst slivers are removed, but some significantly low dihedral angles persist) [24].

By comparison, approaches that start with a background lattice are presented with the opposite challenge. In the absence of a boundary constraint, it may be possible to mesh volumetrically and satisfy a broad range of constraints. However, proving one can still capture the boundary becomes more complex. Typically, these approaches use a highly structured lattice to rapidly construct meshes with are self-similar, such as an octree [92, 116]. While what we can prove about lattice-based algorithms is limited, there is a growing body of work. For 2D domains, it was shown that a quadtree can be provably adapted to lead to a capturing a polyhedral domain [10]. More recent approaches apply the body-centered cubic (BCC) lattice, which is not only a naturally good domain for approximating trivariate functions [54], but also a Delaunay triangulation with good dihedral angles everywhere. Labelle and Shewchuk were the first to use BCC lattices as background meshes to build tetrahedral meshes that conforming to a smooth boundary while maintaining dihedral angle bounds [62]. Algorithms such as those of Zhang et al. [120], Chernikov and Chrisochoides [31], and Liu et al. [66] extend some of these ideas to the case of multimaterial medical domains with significant experimental results. Most recently, Liang and Zhang prove bounds on constructing adaptive meshes which conform to smooth surfaces [64]. Bronson et al. [19] generalize the results of Labelle and Shewchuk with their lattice cleaving approach, and were also able to generalize a proof that in the case of multimaterial boundaries, a bound for the dihedral angles of the resulting elements exists. In this chapter, we generalize the result of lattice cleaving to arbitrary background grids with a broader range of input characteristics.

5.2 Meshing Pipeline

The strategy we propose is to separate the creation and adaptivity of quality volume elements from surface conforming constraints. This separation can be achieved through a volumetric meshing pipeline (Figure 5.1). First, the desired and necessary element characteristics throughout the volume must be determined. These constraints are then used as input to a meshing algorithm to generate an ambient *background* mesh. This mesh will know nothing of material interfaces, but will have appropriately sized elements

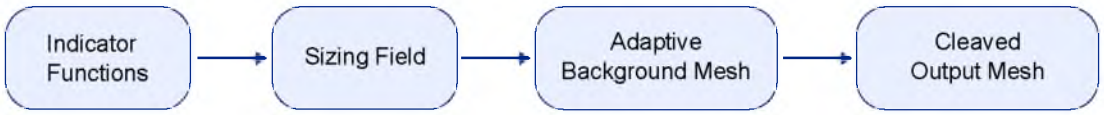


Figure 5.1. Proposed meshing pipeline for conforming volumetric meshing.

thanks to the sizing field driving it. Finally, this background mesh will be fed to the cleaving algorithm, where elements near material interfaces will be cleaved to conform to these surfaces. We provide algorithms which are separately capable of handling the specific pieces of this pipeline.

5.3 Local Feature Size and Sizing Fields

Mesh elements must be adaptively sized for the purpose of geometric fidelity and PDE solution accuracy while simultaneously reducing the number of elements needed for an accurate numerical simulation. A *sizing field* is a scalar field that at every point dictates the ideal size of an element centered around that point. We suggest that a suitable sizing field should posses the following properties: a) it should be small near thin features and high-curvature regions; b) it should progressively increase for larger features and lower-curvature regions; c) it should be sufficiently large at points that are far from material interfaces; and d) it should satisfy Lipschitz continuity conditions. Abrupt changes in the sizing field is undesirable because the quality of the resulting elements is likely to be poor.

In surface mesh generation algorithms, the concept of *feature size* has been widely used to accurately capture the topology of the object that is being meshed. It is defined only on the surface of the object, and it is defined as the distance from the *medial surface* of the object. The medial surface is a surface formed by those points that have more than one closest points on the object boundary. It is also referred to as the "skeletonization" an object. Thin features have a small feature size, and large features have a large feature size. The feature size is used to place vertices on the object surface so that the features are accurately captured by a surface meshing algorithm.

The feature size defined on the surface of an object can be extended over the whole domain to dictate the size of elements in every region of the domain. Persson [82] describes an algorithm that uses a variant of *distance transform* (DT) computed from the surface of the object using the feature size as the initial set of values for the DT. The DT is a scalar field that specifies the distance to the closest point, curve, or surface. It can be visualized as a series of wave fronts emanating from the given set of points, curves, or surfaces. Figure 5.2 provides an illustration of the DT of a C-shaped object and its medial surface (medial axis,

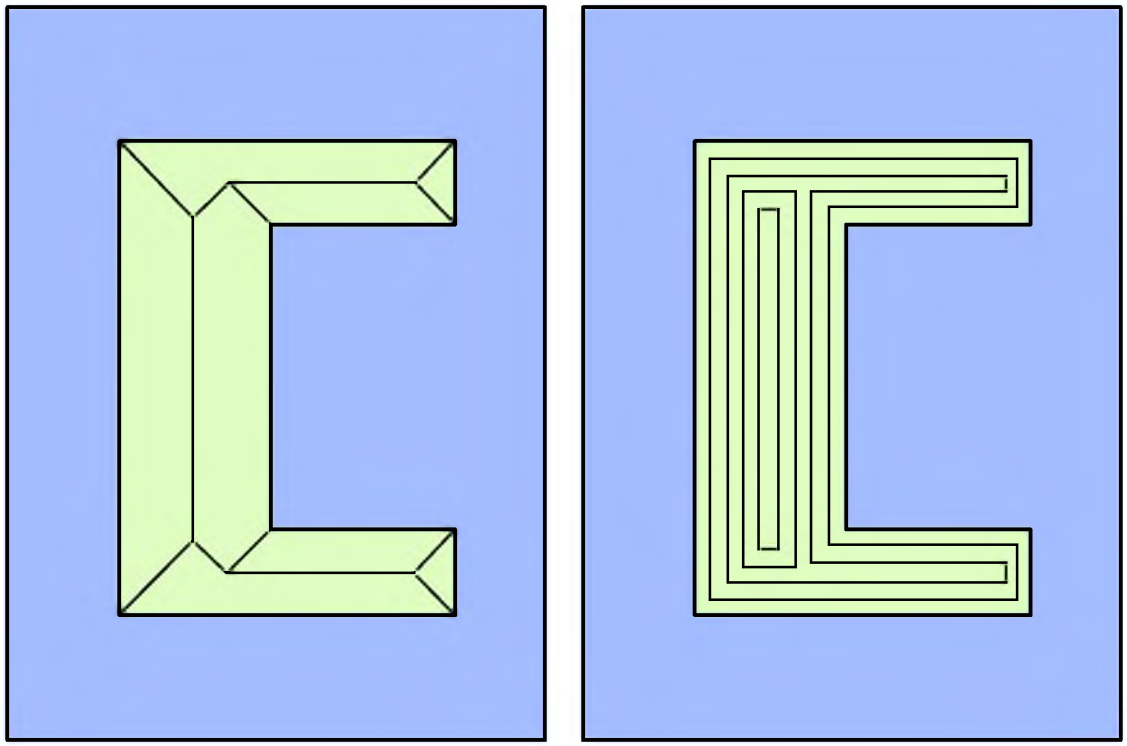


Figure 5.2. The relationship between a shape's medial surface and distance transform level sets. Left: The medial surface (axis) of a C-shaped object. Right: The distance transform level sets. Note, the distance transform is also computed outside the object.

in 2D). Notice that the discontinuity in the Hessian of the DT indicates the medial surface.

The algorithm to compute a sizing field is based on the work of [15] and [82], and it is performed on a voxel domain with subvoxel accuracy. The technique relies on solving for three DTs over this domain. First, the DT is computed starting from the material boundary surfaces. Because the DT is nonsmooth only where the wave fronts collide and is linear otherwise, we can use the Hessian to compute the set of points on the medial surface. The Hessian vanishes at all locations except at the medial surface. The voxels where the Hessian does not vanish define the medial surface. Next, the DT is computed again from the medial surface. The values of the DT at the boundary locations thus define the feature size at those points. Finally, the DT is computed again from the boundary. This time, the initial value at the boundary is set as the feature size computed in the previous step and the gradient of the DT is limited as a user parameter. This gives us the sizing field over the whole domain.

To initialize the starting value for the first DT, we have to find the approximate locations of material interfaces on the edges of the voxel grid. The DT solver uses the fast marching method (FMM) [71] that is second-order accurate and is used with a heap-based priority queue. Note that a $3 \times 3 \times 3$ stencil is required to compute the Hessian. Thus, the resolution of the grid should be appropriately chosen to capture the features of the require size.

5.4 Electrostatic Particle Distributions

Particle systems are often used in mesh generation algorithms in order to obtain a distribution of points satisfying certain constraints. For instance, the idea of *centroidal Voronoi diagrams* has been used in several mesh generation algorithms [43] for isotropic point distribution requirements. The concept has been extended for anisotropic metrics [44] as well. Other notable examples of the use of particle systems in mesh generation include the surface sampling technique of Hart *et al.* [53], Yamakawa and Shimada's ellipsoidal bubble packing algorithm [114] and the particle sampling technique of Meyer *et al.* for multimaterial meshing [76].

In each of these techniques, a stable point distribution is achieved by iteratively minimizing an energy function that matches the meshing requirements. These approaches typically distribute particles on material interfaces first, using these surfaces as a constraint for a volumetric meshing algorithm. In contrast, this new approach to adaptive mesh generation distributes particles directly over the whole of the domain, without any regard for material interfaces. These points are provided to a Delaunay tetrahedralization algorithm to build a background mesh suitable for the next stage of the new meshing pipeline.

We propose a new electrostatic particle simulation technique to distribute particles over the whole input domain. A pipeline describing the algorithm for generating an unstructured background mesh is shown in Figure 5.3. Unlike typical electrostatic particle systems, in which particles tend to gather at protrusions and sharp feature, this new approach produces particle distributions that accurately match the input sizing field specification. Particles are given an electrostatic charge as usual, but the domain itself is also given an opposite charge.

This background charge interacts with particles in such a way that a net-zero charge can only be produced when the set of particles precisely matches the desired input sizing field. The particle distribution technique is designed such that the regions of the domain with a small feature size will have a large charge density, while regions with larger feature size will have a smaller charge density. This ensures that greater number of particles congregate in the region where they are desired. Figure 5.4 provides a pictorial illustration of a particle system where more particles are needed in the three circular regions.

The force on a particle in the domain is a sum of both the background charge density, as well as the other particles in the domain. The force due to the background charge density is determined by computing the gradient of the electrostatic potential due to the charge density, and the force due to other particles is computed by summing up the forces from all other particles in the domain. The particles are moved based on the particle-field and particle-particle interactions. When the system converges to a stable equilibrium, the distribution will respect the provided sizing field.

5.4.1 Charge Density

In order to compute the charge density corresponding to an input sizing field, we exploit the relationship between a sizing field and optimal sphere packing density. The optimal sphere packing density (fraction of volume filled by spheres) for a hexagonal close packing is $\eta = \frac{\pi}{3\sqrt{2}}$. Let the volume of the domain be V, and assume a uniform sizing field, l, in the domain. The number of spheres, n, of the radius $l/2$ in the volume is given by

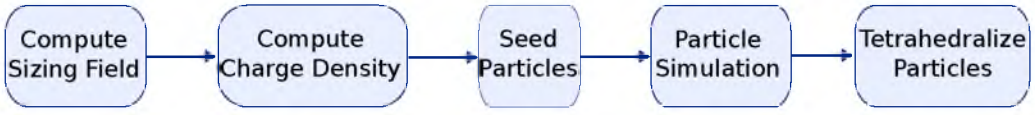


Figure 5.3. Proposed pipeline for generating an adaptive, unstructured background mesh.

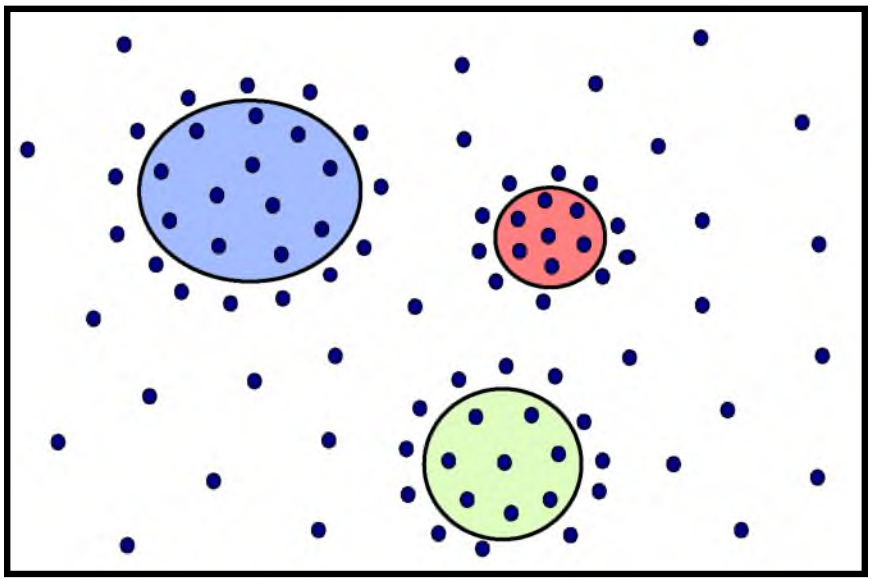


Figure 5.4. An illustration of a particle distribution over a domain. These particles pack with locally desired densities but do not lie directly on any material interfaces.

$$n = \frac{\eta V}{\frac{4}{3}\pi \left(\frac{l}{2}\right)^3}. \tag{5.1}$$

We set the number of particles to be the nearest integer to the total background charge in the domain (computed by integrating the charge density). In this way, we realize a stable equilibrium where negatively-charged particles neutralize the positively-charged background. Therefore, we want $n = \rho V$, where ρ is the charge density. Solving for ρ, we obtain $\rho = \sqrt{2}/l^3$. Thus, the charge density is set to be inversely proportional to the cube of the sizing field. Note that any monotonically decreasing charge density with respect to the sizing field is likely to yield some results that respect the adaptivity requirements, but its feasibility depends upon the application.

5.4.2 Electrostatic Potential

The electrostatic potential due to the background charge density is computed by solving the Poisson equation, $\nabla^2 \mathbf{u}(\mathbf{x}) = f(\mathbf{x})$, where $\mathbf{u}$ is the electrostatic potential and f is the charge density. The Poisson equations are solved using the finite difference scheme on a structured grid with Dirichlet boundary conditions computed by summing up the potential due to the charge density at every point in the boundary. This process is accelerated using an octree-based technique. We use the linear conjugate gradient solver to determine the solution of the linear system resulting from the finite-difference approximation of the equation and boundary conditions.

In order to compute the local charge density and its gradient at any point in the domain, we utilize a cubic convolution-based interpolation technique over the structured grid. Its main advantage over a trilinear interpolation approach is the continuity of the gradient of the interpolated function over the whole domain, which helps the system avoid local minima. Additionally, the analytical gradient of the interpolant can be accurately computed. When trilinear interpolation is used instead, the particles get "trapped" in the faces, edges, and corners of the cubic elements of the grid. This naturally results in a suboptimal point distribution.

5.4.3 Particle Simulation

For the electrostatic simulation, the number of particles that are seeded in the various parts of the domain is proportional to the local charge density. This ensures a quick convergence of the particle system because the movement of particles is locally restricted. Each individual particle is separately moved by a distance proportional to the force on the particle and the step size for that iteration. We adaptively vary the step size in

each iteration. The process is accelerated using the octree-based Barnes-Hut simulation technique [7].

Each octant of the octree stores the location of the center of the charge distribution within the cell, as well as the total contained charge. The force on a particle charge at any location is computed by traversing down the octree. If the ratio of the length of the smallest side of an octant vs. the distance between the particle charge and the center of the charge distribution is lower than a threshold θ, the force due to all the charges in the octant on the particle charge is approximated by a single point. This significantly reduces the number of floating operations required to compute the force on the particle charge. The particles are iteratively moved to optimal locations based on the forces acting upon them. Once a static equilibrium is reached, the particles are tetrahedralized using Tetgen [100].

5.5 Unstructured Cleaving

In order to produce a surface conforming mesh from an unstructured background mesh, we turn to a technique which, up until now, has been demonstrated to work only on regular lattices. Lattice Cleaving, like Isosurface Stuffing, is a stencil-based technique for producing conforming tetrahedral meshes with elements of bounded quality.

In one sense, these algorithms can be considered mesh *processing* algorithms. They take as input a mesh, a regular lattice of high quality tetrahedra (the Body-Centered Cubic (BCC) Lattice), and through a series of vertex warps and stencil operations transform the mesh elements to conform to material boundaries (Figure 5.5). We use this paradigm to extend the technique to arbitrary unstructured and irregular background *input* meshes. We do this in the context of multimaterial volumes. This section details the technical considerations that need to be accounted for to achieve this generalization.

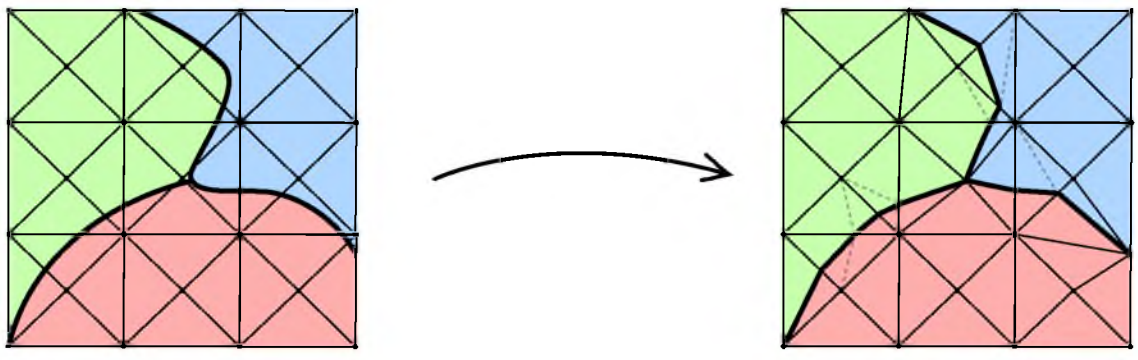


Figure 5.5. Illustration of lattice cleaving in 2D, Left: background mesh with material interfaces overlaid, Right: cleaved output

The basis of the lattice cleaving algorithm remains unchanged. The input elements are still tetrahedra, the violation snapping and warping rules carry over, and the same output stencil set is used. However, there are two fundamental challenges when moving to an unstructured mesh: resolving stencil consistency across the shared face of neighboring background tetrahedra, and alpha parameter selection.

5.5.1 Consistency and Generalization

Some output topologies have multiple permissible tessellations. Without consideration for consistency, two neighboring tetrahedra may stencil their shared face differently, leading to a topological hole in the mesh. This problem can be avoided by carefully orienting each background element before applying output stencils. Isosurface stuffing does this using a simple parity rule that exploits the regular structure of the lattice. As we are interested in unstructured meshes, we cannot rely on any predetermined structure of an input mesh. Instead, we take inspiration from the approach used in the lattice cleaving algorithm.

Rather than rely on the background lattice structure, the lattice cleaving algorithm resolves stencil consistency by taking advantage of the fact that all output stencils can be generated through a series of material collapses of the most complex four-material stencil. This mapping is called a *generalization*. In this model, a set of stencil outputs is consistent if there exists a set of *virtual* interfaces that, when snapped and warped, would have led to that stencil set. The lattice cleaving algorithm for placing these virtual interfaces and their snap destinations is specific to the BCC lattice and so we develop a more versatile algorithm.

We observe that if you take a three-material face, the only way a set of cuts cannot collapse into a simpler stencil form is when their movement forms a cycle (Figure 5.6). We further observe that for any set of valid face generalizations on a tetrahedron, there is always a way to move a virtual quadruple-point to obtain a valid stencil.

One way to guarantee a set of virtual cuts never move in a cycle is to enforce an ordering. If each vertex is given an integer id, then enforcing a rule that a virtual cut always moves to the higher (or lower) vertex is sufficient. Most mesh implementations store an ordered list of vertices anyway, so creating this order is trivial. The remaining work is determining the destination of virtual triple-points and virtual quadruple-points. Because the triple-point of a face is shared, any valid destination will by definition be consistent across the face, and because quadruple-points exist on the interior of tetrahedron, any valid destination will suffice.

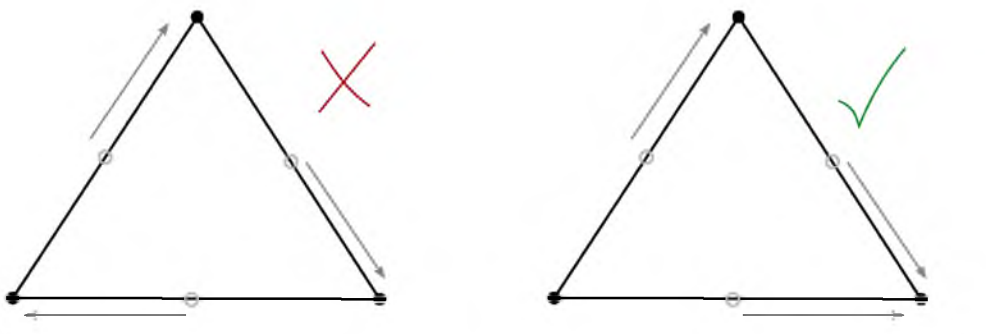


Figure 5.6. Not all virtual cut placements are safe. Left: Cyclic virtual cuts lead to an unsatisfiable generalization. Right: Any ordered priority can lead to safe generalization.

There are two principles to selecting valid virtual interface locations: Virtual interfaces always snap to the next smallest simplex with the most collocated virtual interfaces, and ties are always settled in favor of real interfaces over virtual interfaces. Figure 5.7 illustrates the two ways this manifests on a background lattice face. If one virtual cut exists, the virtual triple snaps to the real cut on the edge incident to the snapped virtual cut. This collapses the missing third material region onto the edge. If two virtual cuts exist, the virtual triple snaps to the colocation of the two virtual cuts (i.e., the vertex with the smallest id). Similarly, the missing materials collapse onto an edge and onto a vertex.

5.5.2 Alpha Selection and Quality Guarantees

In the lattice cleaving and isosurface stuffing algorithms, the alpha parameter controls the trade off between stenciling and warping. It does this by defining the regions in which an interface is considered to be *violating*, and will need to be snapped. Labelle and Shewchuk utilized an automated computational proof to determine optimal alpha parameters for the long and short edges of the BCC lattice in isosurface stuffing. For the multimaterial case,

Figure 5.7. This figure illustrates the generalization of one and two material face stencils. (left) A two-material stencil is generalized. The virtual cut on the edge connecting vertex 0 and vertex 1 moves to vertex 0. The virtual triple follows the cut onto the edge connecting vertex 0 and vertex 2. (right) A one-material stencil is generalized. All virtual cuts move to the adjacent vertices with the lowest index. The virtual triple follows the virtual cuts that end up on the same vertex.

the state space for this proof becomes computational infeasible, and so the authors provide a theoretical proof of bounds, and utilize conservative version of the parameters from the two material case.

As we move to an unstructured mesh, the challenge of finding optimal alpha parameters becomes even more difficult. While the BCC lattice has two edge lengths and symmetric elements, any given unstructured mesh may have no two neighbors with identical edge lengths or element shapes. Therefore, rather than having to choose two alpha values, short and long, the user must pick up to $2|e|$ alpha values. The set of optimal alpha values is therefore unique for every input mesh, and must be computed at run-time.

We present an algorithm for computing conservative alpha violation parameters that allow the quality proof of lattice cleaving to still hold. This algorithm has the benefit of parameterizing the alpha values of each edge by a single global alpha value.

First, we observe that in the limit, as α approaches zero, the background mesh stops all warps and stencil elements can become arbitrarily bad. As α increases, the snapping and warping procedure becomes increasingly aggressive, ultimately to the point where it is unsafe and may result in degenerate elements.

If we permit the four vertices of a tetrahedron to move in any direction, the shortest distance they can travel before the element becomes degenerate (coplanar) is along the shortest vertex altitude. If we show no preference to any particular vertex, they meet in the plane that is halfway along the altitude, or $\frac{h}{2}$ from each vertex, where h is the height of the altitude. Figure 5.8 illustrates this in 2D. This observation provides an upper limit for how aggressive the α selection can be for any particular vertex with respect to a tetrahedron to which it belongs. We can then parameterize over this space as $\alpha = (\frac{1}{2} - \xi)h$ for $0 < \xi < \frac{1}{2}$ and provide ξ as a user parameter to optimize.

The algorithm for computing a set of α parameters is thus as follows: Begin with current best guess for ξ. For each vertex v_i, iterate over all incident tetrahedra and set $\alpha_i = \min\left\{\alpha_i, (\frac{1}{2} - \xi)h\right\}$.

In this formulation, the α parameters around a vertex are all the same and are stored on the vertex, rather than on the edge. This algorithm can be used in conjunction with the proof of bounds from the Lattice Cleaving algorithm [19] to prove that it also places bounds on the quality of output elements. We replace Lemma 3 of the proof with the following alternate lemma.

Lemma 3 *For every tetrahedron with ϵ-good dihedral angles, there exists a space of permissible violation parameters $\bar{\alpha}$ such that the tetrahedron will retain ϵ-good angles after*

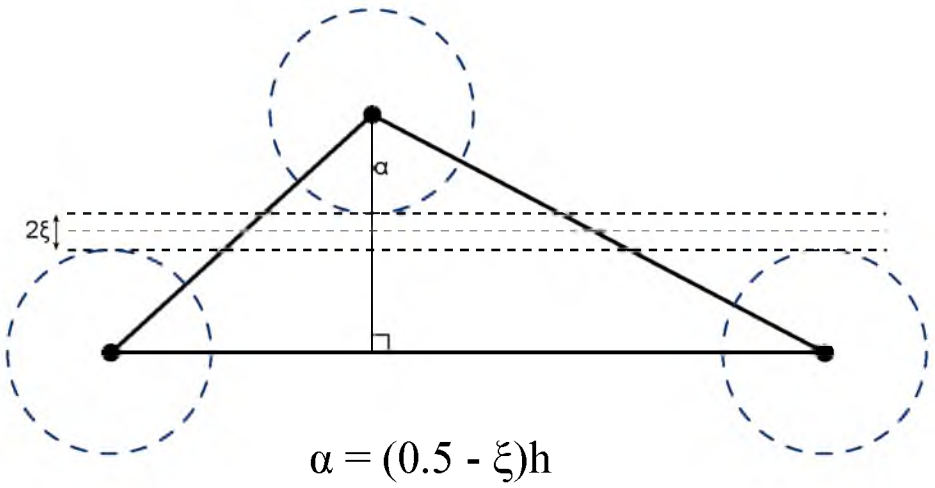


Figure 5.8. Illustration of how the maximum safe distance that a vertex may move is bounded by the height of the corresponding altitude.

warping.

Proof : Let t be a background tetrahedron with ϵ-good angles. Whether measured by aspect ratio or dihedral angle, the tetrahedron decreases in quality towards degeneracy as the vertices approach becoming coplanar. The shortest path vertices can move during a warp to create such a coplanarity is along the shortest altitude. Therefore, any safe set of α values must follow the inequality

$$\alpha_1 + \max\{\alpha_2, \alpha_3, \alpha_4\} < h, \tag{5.2}$$

where α_i is an α-ball around vertex v_i and v_1 is the vertex with the smallest altitude. This inequality is easy to satisfy and can be parameterized as

$$\alpha = (\frac{1}{2} - \xi)h \text{ for } 0 < \xi < \frac{1}{2}. \tag{5.3}$$

As ξ approaches $\frac{1}{2}$, the maximum safe *alpha* values are reached. As ξ approaches 0, warping becomes increasingly restricted.

5.6 Evaluation

Together, the contributions of this chapter offer a full multimaterial volumetric meshing pipeline. In this section, we illustrate what such a system is capable of through the use of both synthetic and real-world data. As discussed in Section 5.4, the electrostatic particle formulation, while extendable to further optimization, currently only optimizes vertex positions and may produce low volume elements. For comparison, for each example

dataset, we generate results using both the electrostatic background mesh, as well as a structured, adaptive octree mesh. In this way, the cleaving of these meshes may more easily be seen as a mesh processing procedure, with results for both input meshes compared side by side.

The implementation for creating the adaptive octree background meshes is straightforward. The octree begins with a single cell that encloses the whole domain. The algorithm queries a sizing field oracle that returns the minimum sizing field within the cell. If this size is smaller than the width of the cell, the tree subdivides. This routine is ran recursively until the smallest local feature size is no smaller than half a cell. Then, the graded stencil set from [19] and [62] is used to fill in the tree and output the background mesh.

We ran the experiments on a single core of the 16-core AMD Opteron 8360 SE 2.5GHz processor with 96GB of RAM running the openSUSE 11.3 (x86_64) operating system with gcc (SUSE Linux) 4.5.0 20100604 compiler. The size of the mesh background and the output generated by the implementation is reported in Table 5.1. The timing for each stage of the pipeline for structured and unstructured meshes is reported in Table 5.2 and 5.3, respectively. In the particle simulation, the threshold θ (see Section 5.4) was reduced from 1 to 0.25 in 200 iterations and held at 0.25 for the next 100 iterations. The time taken for the first 200 iteration is roughly one-third the time reported in Table 5.3.

Figure 5.9 contains a synthetic dataset of various spheres. These spheres can each be independently meshed with a fewer number of tetrahedra, but together produce small cavities that drive the sizing field down. The surfaces and background meshes adapt to this sizing field as necessary. The left result is generated from a graded octree background mesh, and the right result is generated from a graded electrostatic mesh.

Table 5.1. The size of the domain and the sizes of the background and output meshes. S denotes structured meshes, and U denotes unstructured meshes.

Domain	Size	Mesh Type	Background Mesh		Output Mesh	
			#Vertices	# Elements	#Vertices	# Elements
Torus	[64, 64, 64]	S	35,082	175,456	35,951	180,238
		U	13,556	82,298	14,259	85,626
Spheres	[64, 64, 64]	S	26,219	129,804	26,921	133,858
		U	11,856	71,160	12,500	74,050
Torso	[64, 64, 64]	S	145,240	721,678	149,818	746,955
		U	60,867	360,182	64,826	378,361
Frog	[260, 245, 150]	S	1,057,586	5,347,544	1,087,272	5,515,823
		U	70,415	428,173	74,489	447,741

Table 5.2. The time taken for each stage of the pipeline to generate the structured meshes.

Domain	Time (in seconds)		
	Sizing Field	Background Mesh	Warping/Cleaving
Torus	2.15	6	15
Spheres	2.39	7	10
Torso	4.50	43	97
Frog	288.9	520	202

Table 5.3. The time taken for each stage of the pipeline to generate the unstructured meshes. Note the time taken to generate the sizing field is not included in the table because it has already been included in Table 2 for structured meshes.

Domain	Time (in seconds)			
	Poisson Solver	Particle System	Tetgen	Warping/Cleaving
Torus	22	2,466	1.40	5
Spheres	21	2,103	1.30	5
Torso	22	17,727	2.24	45
Frog	1,742	9,474	2.75	50

Figure 5.10 contains two meshes generated from a synthetic dataset of a torus and a sphere in close proximity. Grading along the surfaces as well as the background mesh can be seen in both the structured and unstructured meshes.

Figure 5.11 contains two meshes generated from an MRI of a human torso. The clear cut of the boundary of the domain can be seen on both the octree and electrostatic background meshes. Because the sizing field goes to zero at three-material junctions, user settings limit the sizing field in these regions. The time taken for each stage of pipeline is reasonable for the size of the domain and the size of the mesh needed to capture all the features in the domain.

Figure 5.12 contains two meshes generated from an MRI scan of a frog. The structured version of this mesh ends up being much larger due to the octree dimensions enforcing power-of-two dimensions. The unstructured background mesh has no such requirements. Note that the time taken to compute the sizing field is much larger for this domain than for other domains. This is because the size of the domain and the corresponding large grid on which the distance transforms are computed. The time taken to solve the Poisson equation is also large for the same reason. This can be accelerated using a suitable preconditioner such as the algebraic multigrid preconditioner. Although this domain needs more vertices to

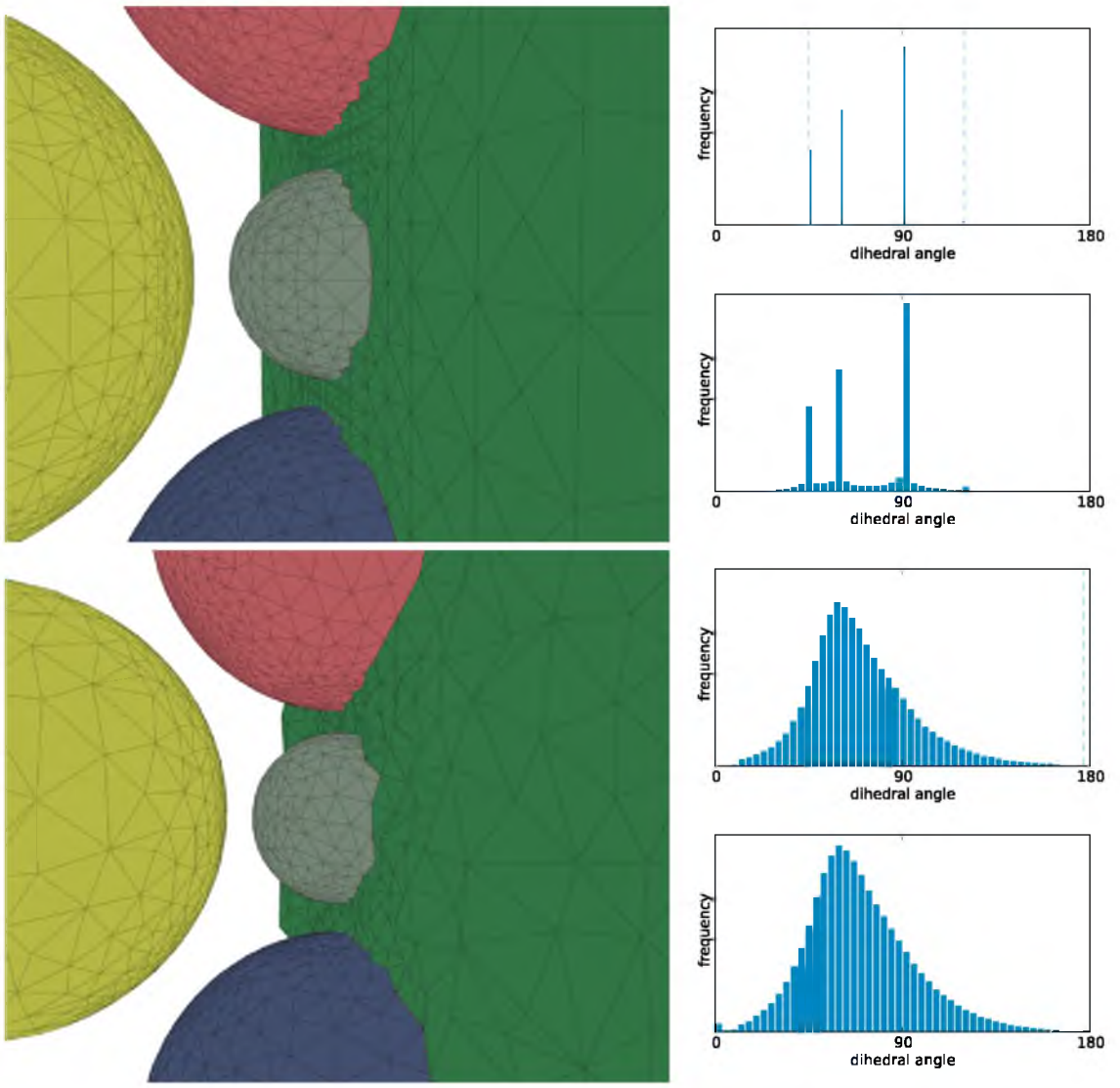


Figure 5.9. A set of four sphere materials. (top) structured background mesh (bottom) unstructured background mesh.

capture all the features present in the data than the torso dataset, the time taken to execute the accelerated particle system is much lower. This is due to the large size of the domain. As the particles are distributed further away from each other, the Barnes-Hut algorithm is able to approximate the forces from groups of particles that are at large distances from a particle in consideration.

Because the particle distribution technique is a global technique, i.e., the position of a particle depends on the position of all other particles (not just its neighbors), it is less likely to get stuck in a local optima. In each of the test cases, we were able to control of the number of vertices in a given region of a domain is a precise manner. While other refinement or

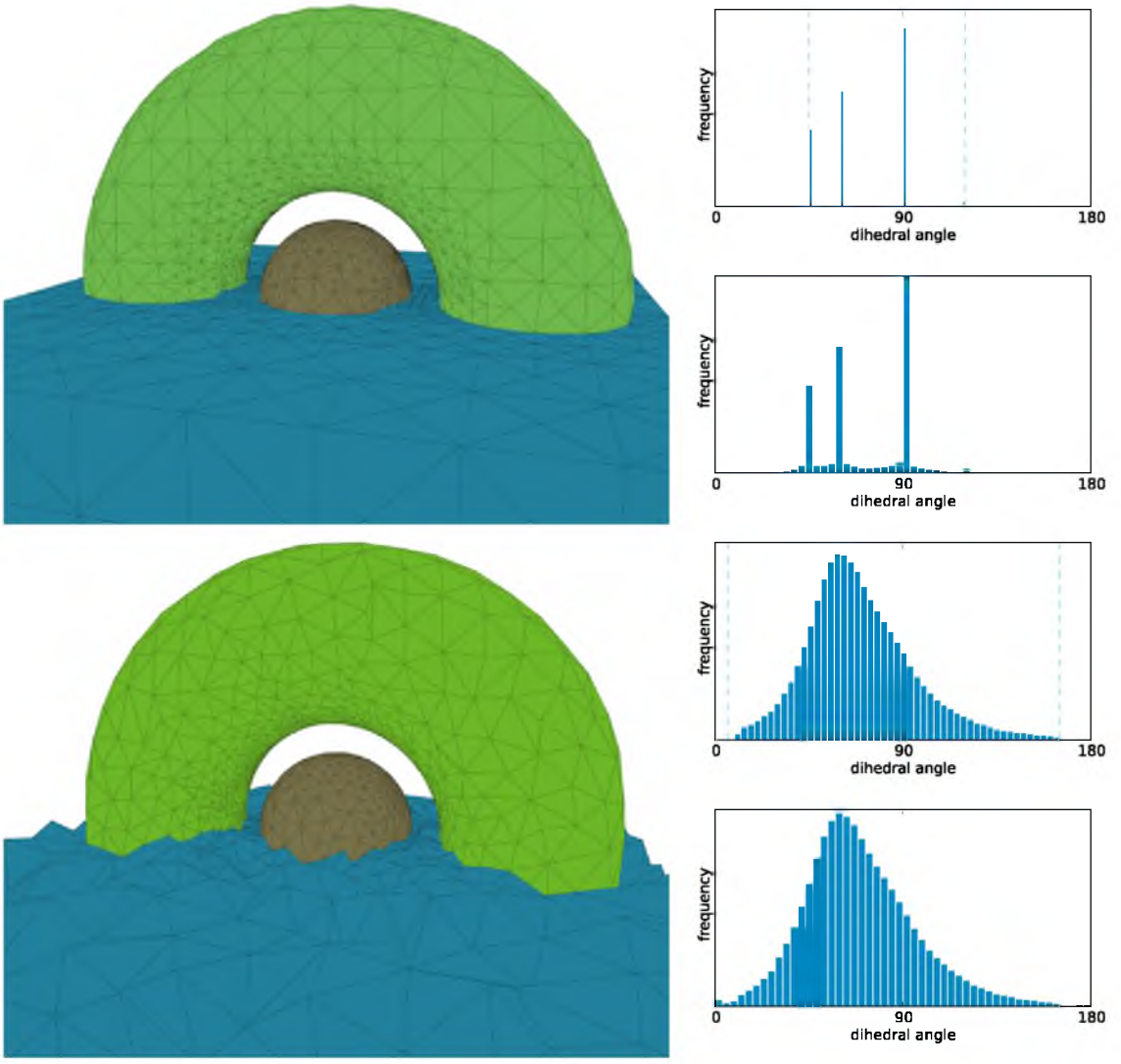


Figure 5.10. A torus with a sphere inside it. (top) structured background mesh (bottom) unstructured background mesh.

variational techniques incrementally add vertices, this technique can determine the number of particles *a priori*. As it is slow, in real applications, we recommend running only a few iterations of the particle distribution scheme and using local techniques to optimize the positions of the particle or to improve the quality of the resulting background mesh.

5.7 Discussion

In this chapter, we have illustrated the potential power of decoupling the problems of mesh element and boundary constraints. The particle system presented achieves its goals with simplicity due to the lack of interface surfaces, and is simply one method for generating

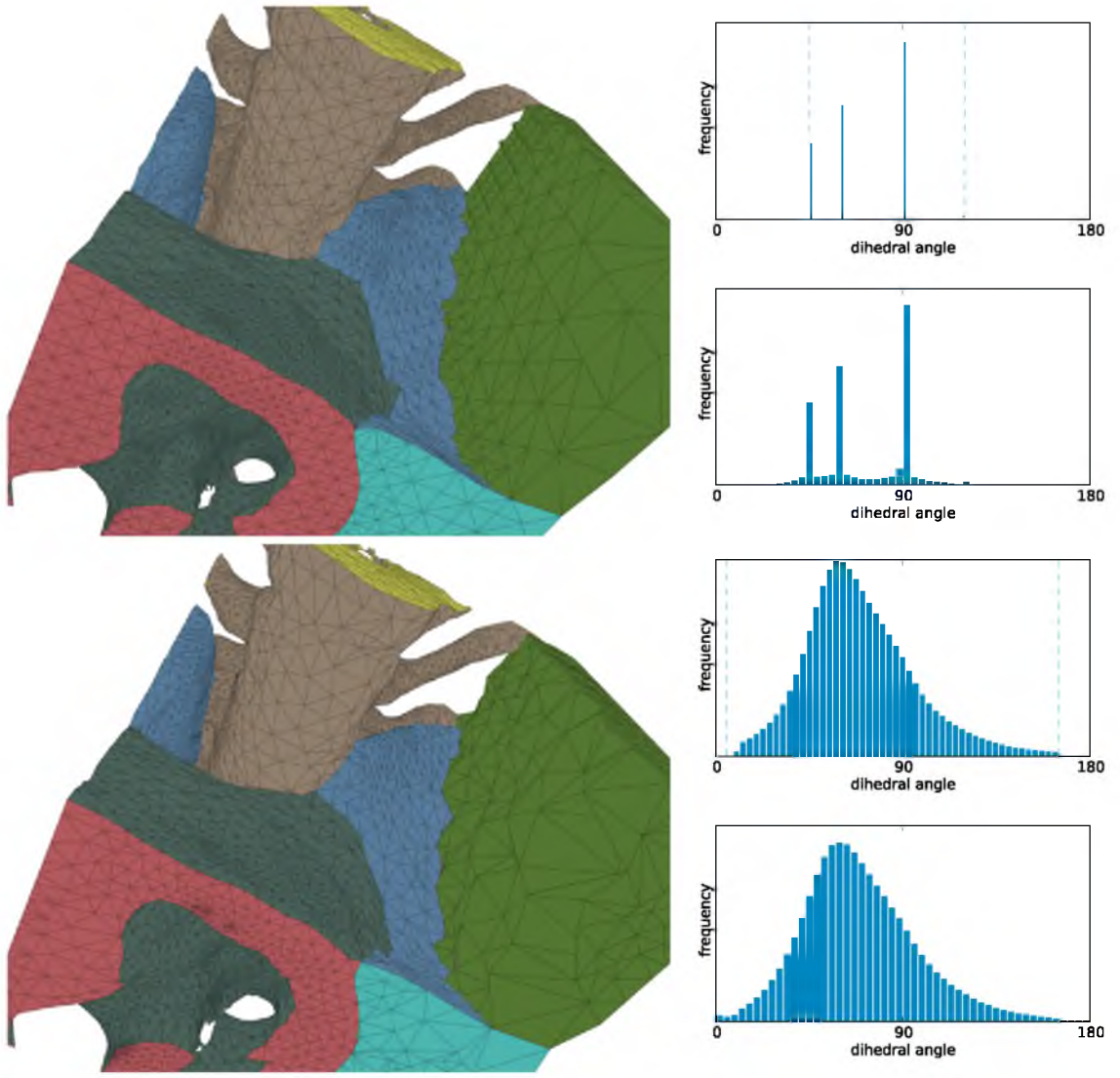


Figure 5.11. Section of human torso MRI with (top) structured and (bottom) unstructured backgrounds meshes.

unstructured background meshes.

One possible alternative algorithm to use for the background mesh generation of the pipeline is centroidal Voronoi tessellation (CVT). This technique is interesting because with variable metrics (as presented in [43]), it has the ability to push low volume elements to the boundary of the domain. This is ideal for the cleaving algorithm, because these external elements will ultimately be discarded.

We have also demonstrated that the lattice cleaving algorithm is extendable to unstructured meshes in a straightforward manner. The method we chose for producing safe α parameters, though conservative, provides a starting point for more sophisticated methods.

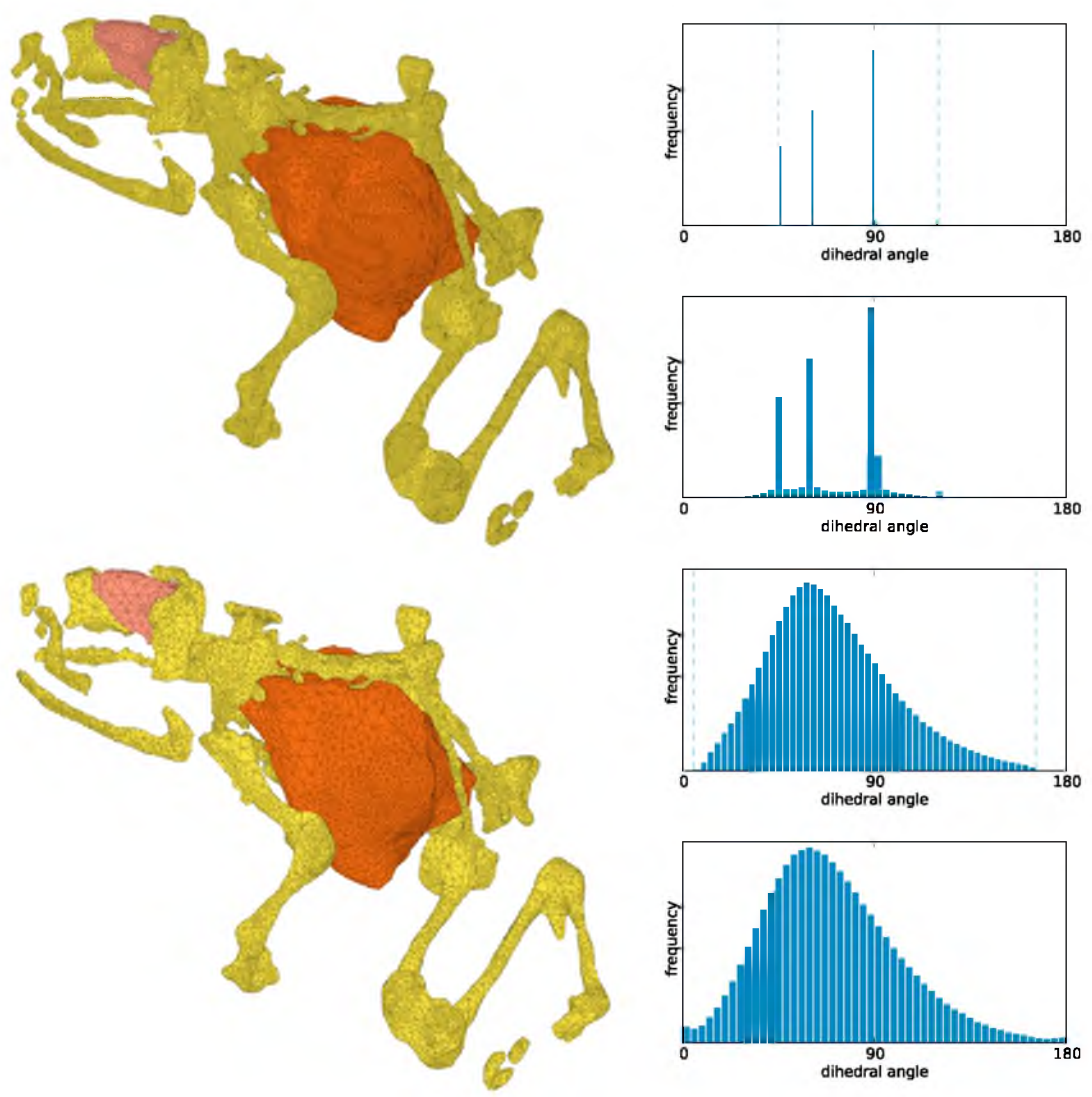


Figure 5.12. An MRI of a frog with (top) structured and (bottom) unstructured background meshes.

Its major drawback is being symmetric around a vertex. This means that as the difference in relative size of neighbor tetrahedra increases, the parameters will be increasingly more conservative. Detaching this symmetry constraint would be a significant step towards solving for truly optimal α parameters.

As a whole, this work suggests that the union of traditional and combinatoric meshing techniques promises to provide a fertile ground for new developments in high-quality conforming mesh generation for unstructured meshes.

CHAPTER 6

DISCUSSION AND FUTURE WORK

This dissertation has presented several novel advances in the area of tetrahedral mesh generation. The particle system from Chapter 3 demonstrated how parametric models are particularly well suited for variational meshing techniques. Optimization and triangulation in parameter space is efficient and does not suffer the complications of working in 3D. The work also showed that a precomputed sizing field is not a requirement for achieving high-quality graded meshes that adapt to feature size. The strategy of inferring a sizing field during optimization adds little cost to the variational approach while still achieving adequate adaptation.

The largest contributions of this work are the lattice and unstructured mesh cleaving algorithms of Chapters 4 and 5. The lattice cleaving algorithm provides the first conforming, multimaterial, tetrahedral meshing algorithm with angle quality guarantees. The speed of the algorithm makes it amenable not only to standard meshing problems, but especially to problems that require iterative remeshing to represent moving geometries. Moreover, the bounds of quality on the lattice cleaving method are significant, enabling numerically well-conditioned simulations. An open-source implementation of this algorithm, called *Cleaver* [90], has been made publicly available, and has been incorporated into several other open-source projects [46, 58, 91].

The generalization of lattice cleaving to unstructured domains is the second largest contribution of this work. Unstructured meshes offer the most flexibility while often requiring fewer elements than structured alternatives. Structured meshes are also unsuitable for some domains such as solid mechanics, where regular structure can introduce bias into solutions. By extending the lattice cleaving algorithm to arbitrary unstructured background meshes, we provided the ability to interchange background meshing algorithms for the desired purpose, while ensuring that these meshes eventually conform to boundary domain without creating degenerate elements.

The methods developed in this paper are by no means complete and leave behind several

new open research problems. The remainder of this final chapter discusses various issues and shortcomings of the methods, with an eye toward future work. It concludes with some previously unpublished early work towards improving topological robustness of the lattice cleaving algorithm.

6.1 Discussion

Looking back at the work in this dissertation, there are a number of shortcomings and areas where improvements are desirable. Some of these shortcomings manifest as necessary trade-offs required for the algorithms to work, while others represent good candidates for further investigation. In this section, we reflect on some of these issues and discuss potential directions of research for addressing them.

In Chapter 5, we presented a proof that lattice cleaving generates bounded quality elements. Unfortunately, this proof does not tell us what the bound actually is, nor what the optimal values of alpha happen to be. The values we used to produce an empirical bound were found through trial and error. While a full computational proof of the style produced for isosurface stuffing appears intractable, an automated routine for pushing the empirical bound would be a highly attractive addition to this work. Conversely, our theoretical proof hints that there may be a noncomputational method of proving the angle bounds for the simpler case of isosurface stuffing. Such a proof would be not only be a great contribution in and of itself, but would also serve as a starting point towards a similar proof for lattice cleaving.

This dissertation focused on tetrahedral mesh generation. However, hexahedra are also a popular finite element for numerical methods. Looking at the success of lattice cleaving, it appears plausible that a similar strategy could be taken for achieving guaranteed quality hexahedral meshes of multimaterial domains. One of the biggest challenges in transferring this technique over to hexahedra is grappling with the space of topologies that can be produced by eight uniquely labeled vertices of a hexahedron. Even the two-material case has confounding many marching cubes implementations with ambiguous topological cases. A technique for automatically enumerating the space of topologies would be essential to tackling this problem.

The biggest shortcoming of the unstructured cleaving algorithm of Chapter 6 is the alpha value selection algorithm. This algorithm was instrumental in proving that the technique produces bounded quality output meshes, but leaves much to be desired. Aside from generating a conservative choice of alpha values, the method forces an unnecessary symmetry on alpha values around vertices. This symmetry corresponds well to the idea of the enclosing

ball used in the proof, but it places an artificial and unnecessary restriction on alpha values. The true space of potential alpha values permits a unique alpha for every vertex-edge pair. This leaves huge potential for improved heuristics, approximations, and algorithms for generating alpha values. Even modest improvement to these routines could yield significant gains in output quality bounds for any given mesh. Rather than fixing alpha values prior to cleaving, one could also imagine a mesh-improvement algorithm that alternates between a cleaving step and an alpha improvement step. A cleaving implementation that supports reversible operations could even selectively optimize the poorest quality regions of any given mesh, preventing the waste of resources on portions of the mesh which, by good fortune, require no improvement.

As shown in the results of Chapter 6, a poor-quality input mesh leaves little room for cleaving to produce an adequate mesh. While the electrostatic particle formulation we used to generate the unstructured background meshes reliably produces point distributions that adapt to feature size, we ultimately rely on the problematic 3D Delaunay tetrahedralization of these points. In order for our method to robustly produce high-quality elements for the background mesh, some measure of volume or tetrahedral quality needs to be incorporated into the solution. If a single formulation is too problematic, a scheme that alternates between optimizing for electrostatic charge and tetrahedral quality might prove sufficient.

On the subject of feature size, our strategy for computing local feature size from a multimaterial field of indicator functions is not particularly robust. Detecting discontinuities in the distance field is a numerically sensitive operation. Further research into robust feature size computation would be indispensable not only to the work of this dissertation, but to every adaptive mesh generation technique that utilizes a sizing field. On the other hand, we ultimately use this sizing field to drive the creation of our background mesh via a particle system. In Chapter 4, we also showed that particle systems can use information passing to infer feature size. This strategy could possibly be adapted to work with our electrostatic formulation. One of the challenges of adapting the same strategy is that unlike the parametric models of Chapter 4, which provided explicit structure as a k-complex, the cleaving algorithm operates over volumetric indicator functions. The topology of the material interfaces defined by these functions is implicit. Overcoming this challenge would be a valuable contribution to this work.

Finally, the mesh cleaving algorithms presented in Chapters 4 and 5 rely on the simplifying assumption that only a single material crossing appears on any background edge. This simplification makes the number of topological cases tractable and representable with

only a limited number of output stencils. Unfortunately, this simplification also means that both sharp and very small features may not be captured properly, depending on their orientation to the background mesh. This can lead to puckering and other artifacts around three-material interfaces. These artifacts are frustrating in that they are the single largest source of geometric error from the method and yet they are caused by a simplification that is crucial for the algorithm to be well formulated. The next section details some early work towards addressing this problem.

6.2 Future Work

As discussed in Chapter 4, the problem of topological aliasing is partially caused by the orientation of the background mesh with respect to the material interfaces. We have observed that small changes to the input background mesh can reduce and sometimes entirely eliminate these topological inaccuracies. We have experimented with two approaches to making these changes systematically. Both of these approaches rely on detecting when a topological error would be made, and modifying the background mesh either during or after construction to avoid it.

In order to detect a topological error, each edge of the background input mesh with a material interface must be evaluated. If an edge properly represents the underlying topology, only two materials will ever appear on the edge. If three or more materials appear on an edge, there must be intersections of the material indicator functions where these additional materials begin to overtake those which are maximum on the vertices. Thus, a straightforward way to detect if a third material is present on the interior of an edge is to compute all material transition points and evaluate whether or not the maximum material is also maximum on an edge vertex. If it is not, then we have found a transition point where an interior material has become maximum. Figure 6.1 illustrates such an example. Every pair of materials must be evaluated, leading to $\binom{n}{2}$ evaluations, where n is the number of materials.

The first approach to solving this problem is routine subdivision. This is good for structured, octree-based meshes. If an octree cell contains at least one topologically problematic edge, the cell is divided. There is no guarantee that this subdivision will not reintroduce a new topological problem at a smaller scale. However, the amount of visual error will always be reduced. That makes this a cheap and attractive approach for visualization purposes. It has the added advantage that it has no impact on the quality bounds of the output mesh. Figure 6.2 illustrates this procedure on the same sharp corner from Chapter 4.

Figures 6.3 and 6.4 illustrate this procedure on planar and spherical input datasets. In

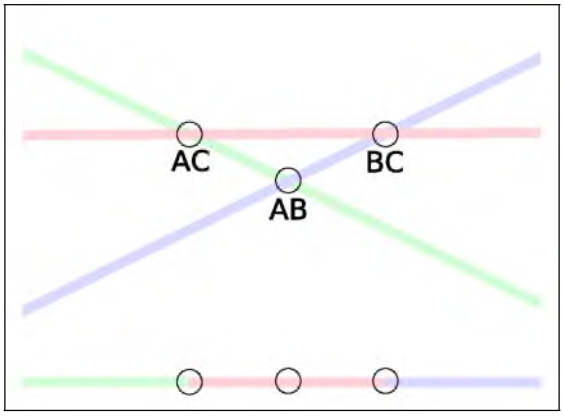


Figure 6.1. An edge with materials a and b maximum on its endpoints, but with a third material c becoming maximum on the interval between. Each pair of material indicator functions is examined for crossings. If the maximum material(s) at a crossing are not also maximum on one of the vertices, then it must be the case that a third material appears on the interior of the simplex.

both cases, the subdivision leads to better accuracy along the three-material interfaces at the cost of a higher number of elements.

The second approach is to subdivide a background tetrahedron precisely where necessary to eliminate topological problems. The idea is to label topologically incorrect edges, and use these labels to output a set of stencil tetrahedra that subdivide missed topologies into a set of smaller more well-behaved background elements. The lattice cleaving algorithm can be hijacked to perform this operation simply by changing the definition of a material interface. Therefore, it seems fitting that we call this approach *topological cleaving*.

Figure 6.5 illustrates this procedure on the same sharp corner. This strategy is diametrically opposed to the previous one. It guarantees that after each iteration, the number of topologically misrepresented materials will be reduced by one. However, it does this at the expense of element quality. This trade-off is partially a bi-product of the fact that material interfaces can meet at arbitrarily sharp angles. Figure 6.6 shows the procedure operating on a corner in 3D. Snapping and warping is disabled to better illustrate the cleaving.

Figure 6.7 shows a comparison of the two approaches within an octree cell of a more problematic dataset. The topological cleaving is able to resolve the missed interfaces in only a single iteration. However, multiple iterations of subdivision only push the problem to a smaller resolution.

These two approaches are both easy to implement and practical, but are not perfect. Ultimately, any method that aims to accurately capture these pathological geometries will have to choose some metric to sacrifice. The subdivision is safe in the sense that it does

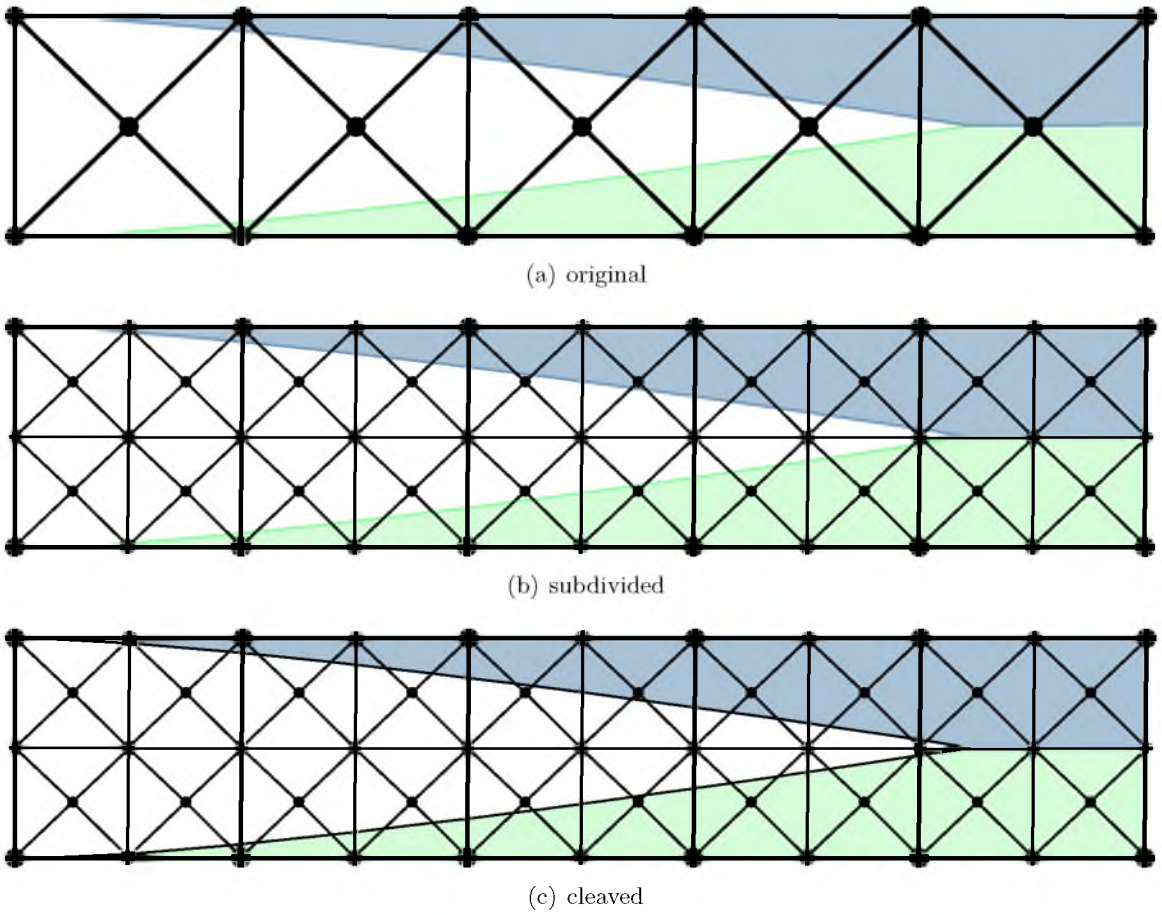


Figure 6.2. The interaction of sharp features on the background lattice. (a) Three materials meet in a sharp corner feature. (b) Each background cell with multiple crossings per edges is subdivided. (c) The subdivided background mesh is cleaved and better resolves the topology. Error has been pushed below the resolution of one cell. Snapping and warping have been skipped to better illustrate the algorithm.

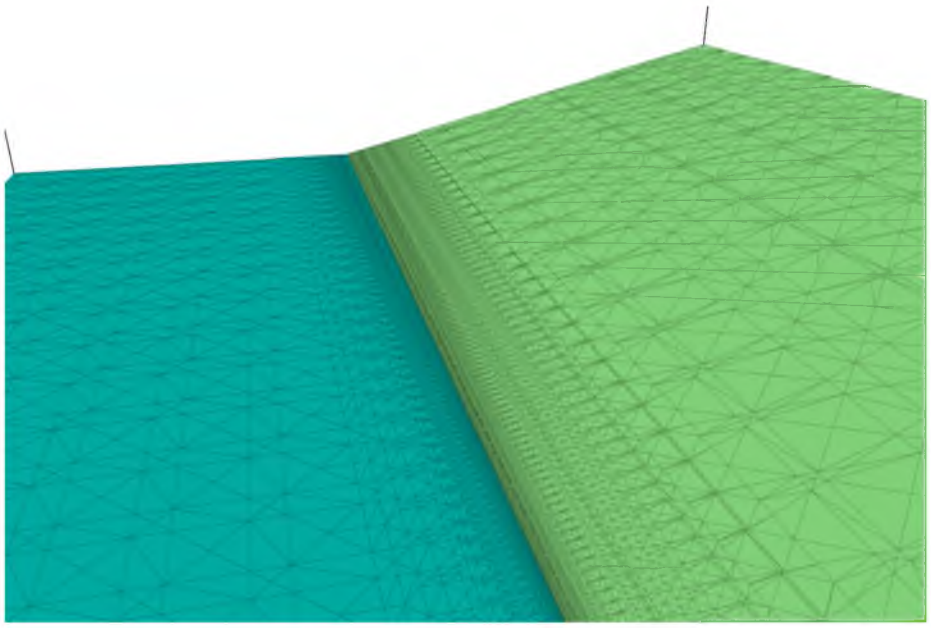


Figure 6.3. Three planar material interfaces meet at a sharp angle. Octree subdivision leads to over-refinement along the three-material junction.

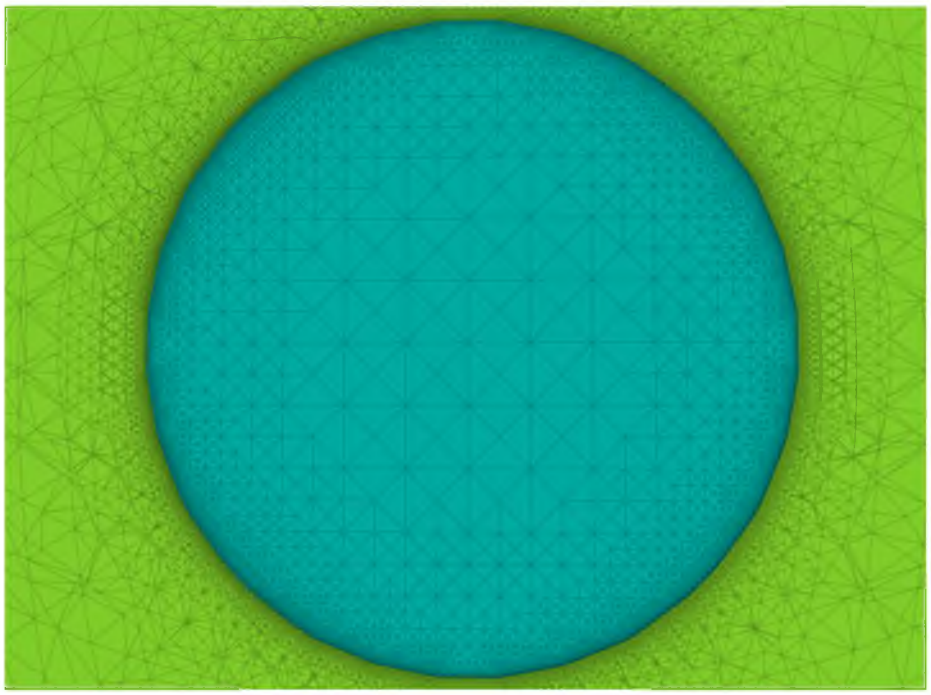


Figure 6.4. Two sphere materials intersect to form an ellipse. Octree subdivision leads to over-refinement along the three-material junction.

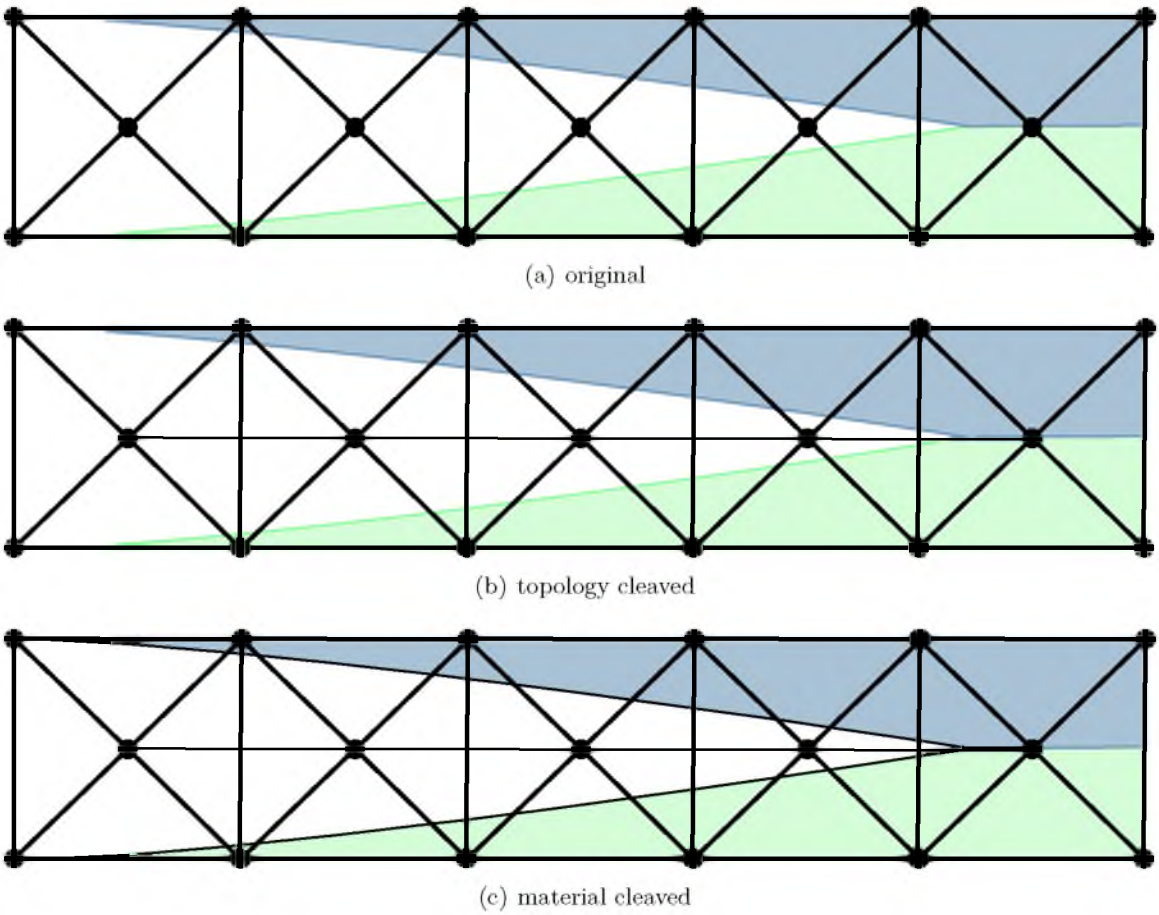


Figure 6.5. Topological cleaving can be used to eliminate aliasing. (a) Three materials meet in a sharp corner feature. (b) Background tetrahedra with multiple crossings are cleaved along the virtual material interface. (c) The altered background mesh is now cleaved to capture true material interfaces. Snapping and warping have been skipped to better illustrate the algorithm.

not lower the output mesh quality bounds. However, it can greatly increase the element count and is not guaranteed to completely fix the topology, only push errors down. On the other hand, the topological cleaving is guaranteed to resolve topology, because each iteration removes precisely one topological ambiguity. It does this at the cost of output mesh quality. Any strategy that could provably fix topological problems while preserving angle bound guarantees would be a valuable extension to this line of work.

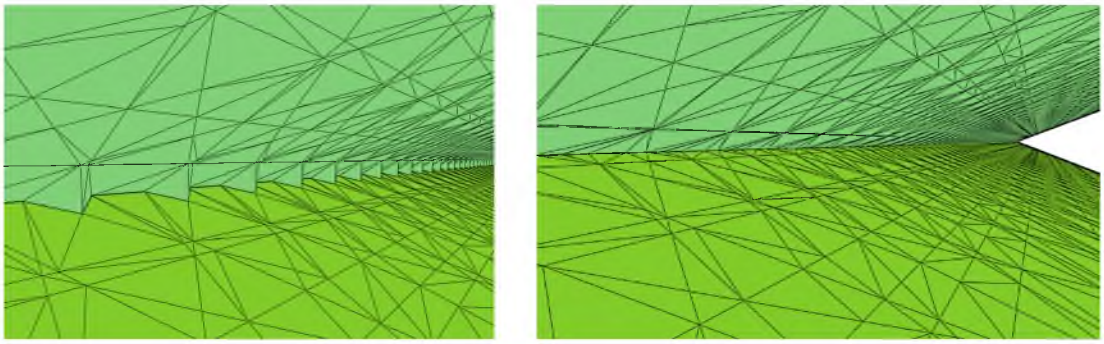


Figure 6.6. Surface view of a sharp corner before and after topological cleaving. (left) A corner formed by three materials suffers from topological logical aliasing. (right) Topological Cleaving corrects the aliasing issue in one pass.

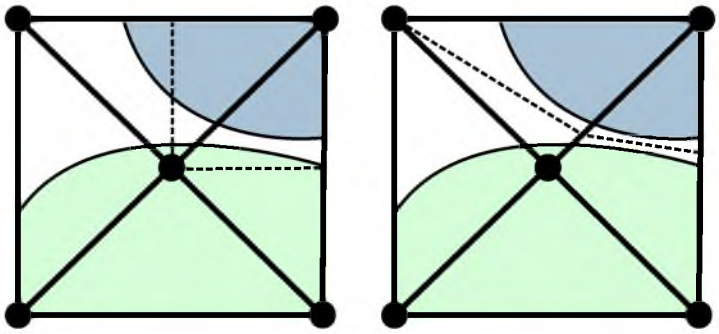


Figure 6.7. This illustration shows the difference between refinement and topological cleaving on a more problematic interface. (left) In this case, cleaving can resolve the topological problem in a single iteration. (right) However, the grid refinement can subdivide multiple times and still not properly capture the topology.

REFERENCES

[1] ALLIEZ, P., COHEN-STEINER, D., YVINEC, M., AND DESBRUN, M. Variational tetrahedral meshing. *ACM Transactions on Graphics 24*, 3 (2005), 617–625.

[2] AMENTA, N., AND BERN, M. W. Surface reconstruction by Voronoi filtering. *Discrete and Computational Geometry 22*, 4 (1999), 481–504.

[3] AMENTA, N., CHOI, S., DEY, T. K., AND LEEKHA, N. A simple algorithm for homeomorphic surface reconstruction. *International Journal of Computational Geometry 12*, 1-2 (2002).

[4] ANTANI, L., DELAGE, C., AND ALLIEZ, P. Mesh sizing with additively weighted Voronoi diagrams. In *16th International Meshing Roundtable* (2007), pp. 335–346.

[5] BABUŠKA, I., AND AZIZ, A. On the angle condition in the finite element method. *SIAM Journal on Numerical Analysis 13*, 2 (1976), 214–226.

[6] BAKER, T. Deformation and quality measures for tetrahedral meshes. In *ECCOMAS* (September 2000).

[7] BARNES, J., AND HUT, P. A hierarchical O(N log N) force-calculation algorithm. *Nature 324* (1986), 446–449.

[8] BENZLEY, S. E., PERRY, E., MERKLEY, K., CLARK, B., AND SJAARDEMA, G. A comparison of all-hexahedral and all-tetrahedral finite element meshes for elastic and elasto-plastic analysis. In *4th International Meshing Roundtable* (1995), pp. 179–191.

[9] BERN, M., CHEW, P., EPPSTEIN, D., AND RUPPERT, J. Dihedral bounds for mesh generation in high dimensions. In *SODA* (1995), pp. 189–196.

[10] BERN, M., EPPSTEIN, D., AND GILBERT, J. Provably good mesh generation. In *Proceedings of Foundations of Computer Science* (1990), IEEE, pp. 231–241.

[11] BIOMESH3D. Quality Mesh Generator for Biomedical Applications. SCI Institute.

[12] BLOOMENTHAL, J., AND FERGUSON, K. Polygonization of non-manifold implicit surfaces. In *SIGGRAPH* (1995), pp. 309–316.

[13] BOISSONNAT, J.-D., AND OUDOT, S. Provably good sampling and meshing of surfaces. *Graphical Models 67*, 5 (2005), 405–451.

[14] BOLTCHEVA, D., YVINEC, M., AND BOISSONNAT, J.-D. Feature preserving Delaunay mesh generation from 3d multi-material images. *Computer Graphics Forum 28*, 5 (2009), 1455–1464.

[15] BOUIX, S., AND SIDDIQI, K. Divergence-based medial surfaces. In *In Sixth European Conference on Computer Vision* (2000), pp. 603–618.

[16] Branets, L., and Carey, G. F. Condition number bounds and mesh quality. *Numerical Linear Algebra with Applications 17*, 5 (2010), 855–869.

[17] Bronson, J., Levine, J., and Whitaker, R. Particle systems for adaptive, isotropic meshing of cad models. *Engineering with Computers 28*, 4 (2012), 331–344.

[18] Bronson, J., Levine, J. A., and Whitaker, R. Lattice cleaving: Conforming tetrahedral meshes of multimaterial domains with bounded quality. In *21st International Meshing Roundtable* (Oct. 2012), pp. 191–210.

[19] Bronson, J., Levine, J. A., and Whitaker, R. Lattice cleaving: A multimaterial tetrahedral meshing algorithm with guarantees. *IEEE Transactions on Visualization and Computer Graphics 20*, 2 (2014), 223–237.

[20] Bronson, J. R., Levine, J. A., and Whitaker, R. T. Particle systems for adaptive, isotropic meshing of CAD models. In *19th International Meshing Roundtable* (Oct. 2010), pp. 279–296.

[21] Bronson, J. R., Sastry, S., Levine, J. A., and Whitaker, R. T. Adaptive and unstructured mesh cleaving. In *23th International Meshing Roundtable* (Oct. 2014).

[22] Chen, L. Mesh smoothing schemes based on optimal Delaunay triangulations. In *13th International Meshing Roundtable* (2004), pp. 109–120.

[23] Cheng, S.-W., and Dey, T. K. Maintaining deforming surface meshes. In *19th Symposium on Discrete Algorithms* (2008), pp. 112–121.

[24] Cheng, S.-W., Dey, T. K., Edelsbrunner, H., Facello, M. A., and Teng, S.-H. Sliver exudation. In *Symposium on Computational Geometry* (1999), pp. 1–13.

[25] Cheng, S.-W., Dey, T. K., and Levine, J. A. A practical Delaunay meshing algorithm for a large class of domains. In *16th International Meshing Roundtable* (2007), pp. 477–494.

[26] Cheng, S.-W., Dey, T. K., and Ramos, E. A. Delaunay refinement for piecewise smooth complexes. *Discrete and Computational Geometry 43*, 1 (2010), 121–166.

[27] Cheng, S.-W., Dey, T. K., Ramos, E. A., and Ray, T. Sampling and meshing a surface with guaranteed topology and geometry. *SIAM Journal on Computing 37*, 4 (2007), 1199–1227.

[28] Cheng, S.-W., Dey, T. K., and Shewchuk, J. R. *Delaunay Mesh Generation*. Chapman and Hall / CRC Computer and information science series. CRC Press, 2013.

[29] Cheng, S.-W., and Poon, S.-H. Three-dimensional Delaunay mesh generation. *Discrete and Computational Geometry 36*, 3 (2006), 419–456.

[30] Chentanez, N., Feldman, B. E., Labelle, F., O'Brien, J. F., and Shewchuk, J. R. Liquid simulation on lattice-based tetrahedral meshes. In *SCA* (Aug. 2007), pp. 219–228.

[31] Chernikov, A. N., and Chrisochoides, N. Multitissue tetrahedral image-to-mesh conversion with guaranteed quality and fidelity. *SIAM Journal on Scientific Computing 33*, 6 (2011), 3491–3508.

[32] Chew, L. P. Constrained delaunay triangulations. In *Proceedings of the third annual symposium on Computational geometry* (New York, NY, USA, 1987), SCG '87, ACM, pp. 215–222.

[33] Chew, L. P. Guaranteed-quality triangular meshes. Tech. Rep. TR-89-983, Computer Science Dept., Cornell University, 1989.

[34] Chew, L. P. Guaranteed-quality mesh generation for curved surfaces. In *Symposium on Computational Geometry* (1993), pp. 274–280.

[35] Cifuentes, A. O., and Kalbag, A. A performance study of tetrahedral and hexahedral elements in 3-D finite element structural analysis. *Finite Elements in Analysis and Design 12*, 3-4 (1992), 313–318.

[36] Coupez, T., Jannoun, G., Veysset, J., and Hachem, E. Edge-based anisotropic mesh adaptation for cfd applications. In *Proceedings of the 21st International Meshing Roundtable*, X. Jiao and J.-C. Weill, Eds. Springer Berlin Heidelberg, 2013, pp. 567–583.

[37] Dannhauer, M., Lanfer, B., Wolters, C. H., and KnÃűsche, T. R. Modeling of the human skull in EEG source analysis. *Human Brain Mapping 32*, 9 (2011), 1383–1399.

[38] D'Azevedo, E. F. Are bilinear quadrilaterals better than linear triangles? *SIAM Journal on Scientific Computing 22*, 1 (2000), 198–217.

[39] Dey, T. K., Janoos, F., and Levine, J. A. Meshing interfaces of multi-label data with Delaunay refinement. *Engineering with Computers 28*, 1 (Jan. 2012), 71–82.

[40] Dey, T. K., Li, G., and Ray, T. Polygonal surface remeshing with Delaunay refinement. In *14th International Meshing Roundtable* (2005), pp. 343–361.

[41] Dey, T. K., and Sun, J. Normal and feature approximations from noisy point clouds. In *Foundations of Software Technology and Theoretical Computer Science* (2006), pp. 21–32.

[42] Drone, S. Real-time particle systems on the gpu in dynamic environments. In *ACM SIGGRAPH 2007 Courses* (New York, NY, USA, 2007), SIGGRAPH '07, ACM, pp. 80–96.

[43] Du, Q., and Wang, D. Tetrahedral mesh generation and optimization based on centroidal Voronoi tessellations. *International Journal for Numerical Methods in Engineering 56*, 9 (2003), 1355–1373.

[44] Du, Q., and Wang, D. Anisotropic centroidal Voronoi tessellations and their applications. *SIAM Journal on Scientific Computing 26*, 3 (2005), 737–761.

[45] Etiene, T., Nonato, L., Scheidegger, C., Tienry, J., Peters, T., Pascucci, V., Kirby, R., and Silva, C. Topology verification for isosurface extraction. *IEEE Transactions on Visualization and Computer Graphics 18*, 6 (2012), 952–965.

[46] Fedorov, A., Beichel, R., Kalpathy-Cramer, J., Finet, J., Fillion-Robin, J.-C., Pujol, S., Bauer, C., Jennings, D., Fennessy, F., Sonka, M., Buatti, J., Aylward, S., Miller, J., Pieper, S., and Kikinis, R. 3d slicer as an image computing platform for the quantitative imaging network. *Magnetic Resonance Imaging 30*, 9 (11 2012), 1323–41.

[47] Freitag, L. A., and Ollivier-Gooch, C. Tetrahedral mesh improvement using swapping and smoothing. *International Journal for Numerical Methods in Engineering 40*, 21 (1997), 3979–4002.

[48] Frey, P., and George, P. *Mesh Generation.* ISTE. Wiley, 2010.

[49] Fuchs, A. Automatic grid generation with almost regular Delaunay tetrahedra. In *International Meshing Roundtable* (1998), pp. 133–147.

[50] Guéziec, A., and Hummel, R. A. Exploiting triangulated surface extraction using tetrahedral decomposition. *IEEE Transactions on Visualization and Computer Graphics 1*, 4 (1995), 328–342.

[51] Haimes, R., and Aftosmis, M. J. On generating high quality watertight triangulations directly from CAD. In *International Society of Grid Generation* (2002).

[52] Haimes, R., and Follen, G. J. Computational analysis programming interface. In *6th International Conference on Numerical Grid Generation in Computational Field Simulation* (1998), pp. 663–672.

[53] Hart, J. C., Bachta, E., Jarosz, W., and Fleury, T. Using particles to sample and control more complex implicit surfaces. In *ACM SIGGRAPH 2005 Courses* (New York, NY, USA, 2005), SIGGRAPH '05, ACM, p. 269.

[54] Hossain, Z., Alim, U. R., and Möller, T. Toward high-quality gradient estimation on regular lattices. *IEEE Transactions on Visualization and Computer Graphics 17*, 4 (2011), 426–439.

[55] Isaacson, B. M., Brunker, L. B., Brown, A. A., Beck, J. P., Burns, G. L., and Bloebaum, R. D. An evaluation of electrical stimulation for improving periprosthetic attachment. *Journal of Biomedical Materials Research Part B: Applied Biomaterials 97B*, 1 (2011), 190–200.

[56] Ju, T., Losasso, F., Schaefer, S., and Warren, J. Dual contouring of hermite data. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 2002), SIGGRAPH '02, ACM, pp. 339–346.

[57] Karkanis, T., and Stewart, A. J. Curvature-dependent triangulation of implicit surfaces. *IEEE Computer Graphics and Applications 21*, 2 (2001), 60–69.

[58] Kitware. Bender: Open source software for repositioning voxelized anatomical models., 2014.

[59] Klingner, B. M., and Shewchuk, J. R. Aggressive tetrahedral mesh improvement. In *16th International Meshing Roundtable* (2007), pp. 3–23.

[60] Knupp, P. Algebraic mesh quality metrics. *SIAM Journal on Scientific Computing 23*, 1 (2001), 193–218.

[61] Labelle, F. Sliver removal by lattice refinement. *International Journal of Computational Geometry and Applications 18*, 06 (2008), 509–532.

[62] Labelle, F., and Shewchuk, J. R. Isosurface stuffing: fast tetrahedral meshes with good dihedral angles. In *SIGGRAPH* (2007).

[63] Leng, J., Zhang, Y., and Xu, G. A novel geometric flow approach for quality improvement of multi-component tetrahedral meshes. *Computer-Aided Design* (2013).

[64] Liang, X., and Zhang, Y. An octree-based dual contouring method for triangular and tetrahedral mesh generation with guaranteed angle range. *Engineering with Computers* (2013). Accepted.

[65] Liu, Y., Foteinos, P., Chernikov, A., and Chrisochoides, N. Multi-tissue mesh generation for brain image. In *19th International Meshing Roundtable* (October 2010), pp. 367–384.

[66] Liu, Y., Foteinos, P., Chernikov, A., and Chrisochoides, N. Mesh deformation-based multi-tissue mesh generation for brain images. *Engineering with Computers 28*, 4 (2012), 305–318.

[67] Lohner, R. Surface gridding from discrete data. In *4th International Meshing Roundtable* (1995), pp. 29–44.

[68] Löhner, R., and Parikh, P. Generation of three-dimensional unstructured grids by the advancing-front method. *International Journal for Numerical Methods in Fluids 8*, 10 (1988).

[69] Lorensen, W. E., and Cline, H. E. Marching cubes: A high resolution 3d surface construction algorithm. In *SIGGRAPH* (1987), ACM, pp. 163–169.

[70] MacLeod, R. S., Stinstra, J. G., Lew, S., Whitaker, R. T., Swenson, D. J., Cole, M. J., KrÃijger, J., Brooks, D. H., and Johnson, C. R. Subject-specific, multiscale simulation of electrophysiology: a software pipeline for image-based models and application examples. *Philosophical transactions. Series A, Mathematical, physical, and engineering sciences 367*, 1896 (2009).

[71] Malladi, R., and Sethian, J. A. Level set and fast marching methods in image processing and computer vision. In *ICIP* (1996), vol. 1, pp. 489–492.

[72] Marcum, D. *Unstructured Grid Generation Using Automatic Point Insertion and Local Reconnection.* CRC Press, Dec. 1998, pp. 18–31.

[73] Merriman, B., Bence, J. K., and Osher, S. J. Motion of multiple junctions: A level set approach. *Journal of Computational Physics 112*, 2 (1994), 334 – 363.

[74] Meyer, M. D. *Dynamic particles for adaptive sampling of implicit surfaces.* PhD thesis, University of Utah, Salt Lake City, UT, USA, 2008.

[75] Meyer, M. D., Kirby, R. M., and Whitaker, R. T. Topology, accuracy, and quality of isosurface meshes using dynamic particles. *IEEE Transactions on Visualization and Computer Graphics 13*, 6 (2007), 1704–1711.

[76] Meyer, M. D., Whitaker, R. T., Kirby, R. M., Ledergerber, C., and Pfister, H. Particle-based sampling and meshing of surfaces in multimaterial volumes. *IEEE Transactions on Visualization and Computer Graphics 14*, 6 (2008), 1539–1546.

[77] Molino, N., Bridson, R., Teran, J., and Fedkiw, R. A crystalline, red green strategy for meshing highly deformable objects with tetrahedra. In *13th International Meshing Roundtable* (2003), pp. 103–114.

[78] Nielson, G. M., and Franke, R. Computing the separating surface for segmented data. In *IEEE Visualization* (1997), pp. 229–233.

[79] Okabe, A., Boots, B., and Sugihara, K. *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams*. John Wiley & Sons, Inc., New York, NY, USA, 1992.

[80] Paige, C., and Saunders, M. Solution of sparse indefinite systems of linear equations. *SIAM Journal on Numerical Analysis 12*, 4 (1975), 617–629.

[81] Pasko, A. A., Adzhiev, V., Sourin, A., and Savchenko, V. V. Function representation in geometric modeling: concepts, implementation and applications. *The Visual Computer 11*, 8 (1995), 429–446.

[82] Persson, P.-O. Mesh size functions for implicit geometries and PDE-based gradient limiting. *Engineering with Computers 22*, 2 (2006), 95–109.

[83] Pons, J.-P., Ségonne, F., Boissonnat, J.-D., Rineau, L., Yvinec, M., and Keriven, R. High-quality consistent meshing of multi-label datasets. In *IPMI* (2007), pp. 198–210.

[84] PÃŁbay, P. P., and Baker, T. J. A comparison of triangle quality measures. In *Proceedings of the National Academy of Sciences* (1991), Plenum Press, pp. 9653–9657.

[85] Rineau, L., and Yvinec, M. Meshing 3d domains bounded by piecewise smooth surfaces. In *16th International Meshing Roundtable* (2007), pp. 443–460.

[86] Ruppert, J. A Delaunay refinement algorithm for quality 2-dimensional mesh generation. *J. Algorithms 18*, 3 (1995), 548–585.

[87] Scheidegger, C., Fleishman, S., and Silva, C. Triangulating point-set surfaces with bounded error. In *Proceedings of the third Eurographics/ACM Symposium on Geometry Processing* (2005), M. Desbrun and H. Pottman, Eds., The Eurographics Association, pp. 63–72.

[88] Schreiner, J., Scheidegger, C., and Silva, C. High quality extraction of isosurfaces from regular and irregular grids. *IEEE Transactions on Visualization and Computer Graphics (Proceedings Visualization / Information Visualization 2006) 12*, 5 (September-October 2006), 1205–1212.

[89] Schreiner, J. M., Scheidegger, C. E., Fleishman, S., and Silva, C. T. Direct (re)meshing for efficient surface processing. *Comput. Graph. Forum 25*, 3 (2006), 527–536.

[90] SCI Institute. Cleaver: A multimaterial tetrahedral meshing library and application, 2012.

[91] SCI Institute. SCIRun: A Scientific Computing Problem Solving Environment, 2012.

[92] Shephard, M. S., and Georges, M. K. Automatic three-dimensional mesh generation by the finite octree technique. *International Journal for Numerical Methods in Engineering 32*, 4 (1991), 709–749.

[93] Shepherd, J. F., and Johnson, C. R. Hexahedral mesh generation constraints. *Engineering with Computers 24*, 3 (2008), 195–213.

[94] Shewchuk, J. *Delaunay Refinement Mesh Generation.* PhD thesis, Carnegie Mellon University, 1997.

[95] Shewchuk, J. R. Triangle: Engineering a 2D quality mesh generator and Delaunay triangulator. In *Applied Computational Geometry: Towards Geometric Engineering*, M. C. Lin and D. Manocha, Eds., vol. 1148. Springer-Verlag, 1996, pp. 203–222.

[96] Shewchuk, J. R. Tetrahedral mesh generation by Delaunay refinement. In *14th Symposium on Computational Geometry* (1998), pp. 86–95.

[97] Shewchuk, J. R. What is a good linear element? interpolation, conditioning, and quality measures. In *International Meshing Roundtable* (2002), pp. 115–126.

[98] Shewchuk, J. R. General-dimensional constrained delaunay and constrained regular triangulations i: Combinatorial properties. In *Discrete and Computational Geometry* (2005).

[99] Shewchuk, J. R. Unstructured mesh generation. In *Combinatorial Scientific Computing*, U. Naumann and O. Schenk, Eds. CRC Press, Jan. 2012, ch. 10, pp. 257–297.

[100] Si, H. TetGen: A Quality Tetrahedral Mesh Generator and Three-Dimensional Delaunay Triangulator. http://tetgen.berlios.de/.

[101] Si, H., and Gärtner, K. Meshing piecewise linear complexes by constrained Delaunay tetrahedralizations. In *14th International Meshing Roundtable* (2005), pp. 147–163.

[102] Staten, M. L., Kerr, R., Owen, S. J., and Blacker, T. D. Unconstrained paving and plastering: Progress update. In *15th International Meshing Roundtable* (2006), pp. 469–486.

[103] Swenson, D. J., Levine, J. A., Fu, Z., Tate, J., and MacLeod, R. S. The effect of non-conformal finite element boundaries on electrical monodomain and bidomain simulations. *Computing in Cardiology 37* (Sept. 2010), 97–101.

[104] CGAL. Computational Geometry Algorithms Library. http://www.cgal.org.

[105] Tournois, J., Alliez, P., and Devillers, O. Interleaving Delaunay refinement and optimization for 2d triangle mesh generation. In *16th International Meshing Roundtable* (2007), pp. 83–101.

[106] Tournois, J., Wormser, C., Alliez, P., and Desbrun, M. Interleaving Delaunay refinement and optimization for practical isotropic tetrahedron mesh generation. *ACM Transactions on Graphics 28*, 3 (2009).

[107] Triedman, J. K., Jolley, M., Stinstra, J., Brooks, D. H., and MacLeod, R. Predictive modeling of defibrillation using hexahedral and tetrahedral finite element models: recent advances. *Journal of Electrocardiology 41*, 6 (2008), 483 – 486.

[108] Turk, G. Re-tiling polygonal surfaces. In *SIGGRAPH '92* (1992), pp. 55–64.

[109] Valette, S., Chassery, J.-M., and Prost, R. Generic remeshing of 3d triangular meshes with metric-dependent discrete Voronoi diagrams. *IEEE Transactions on Visualization and Computer Graphics 14*, 2 (2008), 369–381.

[110] Wang, J., and Yu, Z. Feature-sensitive tetrahedral mesh generation with guaranteed quality. *Computer-Aided Design 44*, 5 (2012), 400 – 412.

[111] Williams, J., and Rossignac, J. Tightening: Curvature-limiting morphological simplification, gvu. Tech. rep., ACM Symposium on Solid and Physical Modeling (SPM), 2004.

[112] Witkin, A. P., and Heckbert, P. S. Using particles to sample and control implicit surfaces. In *SIGGRAPH '94* (1994), pp. 269–277.

[113] Yamakawa, S., and Shimada, K. High quality anisotropic tetrahedral mesh generation via ellipsoidal bubble packing. In *9th International Meshing Roundtable* (2000), pp. 263–274.

[114] Yamakawa, S., and Shimada, K. Anisotropic tetrahedral meshing via bubble packing and advancing front. *International Journal for Numerical Methods in Engineering 57*, 13 (2003), 1923–1942.

[115] Yan, D.-M., Lévy, B., Liu, Y., Sun, F., and Wang, W. Isotropic remeshing with fast and exact computation of restricted Voronoi diagram. *Computer Graphics Forum 28*, 5 (2009), 1445–1454.

[116] Yerry, M. A., and Shephard, M. S. Automatic three-dimensional mesh generation by the modified-octree technique. *International Journal for Numerical Methods in Engineering 20* (1984), 1965–1990.

[117] Yu, Z., Holst, M. J., and McCammon, J. A. High-fidelity geometric modeling for biomedical applications.

[118] Zhang, N., Hong, W., and Kaufman, A. Dual contouring with topology-preserving simplification using enhanced cell representation. In *Proceedings of the conference on Visualization '04* (Washington, DC, USA, 2004), VIS '04, IEEE Computer Society, pp. 505–512.

[119] Zhang, Y., Bajaj, C., and Sohn, B.-S. 3d finite element meshing from imaging data. *Computer Methods in Applied Mechanics and Engineering 194* (2005), 5083 – 5106.

[120] Zhang, Y., Hughes, T., and Bajaj, C. L. Automatic 3D mesh generation for a domain with multiple materials. In *16th International Meshing Roundtable* (2007), pp. 367–386.

[121] Zhang, Y., and Qian, J. Resolving topology ambiguity for multiple-material domains. *Computer Methods in Applied Mechanics and Engineering*, 0 (2012), 166 – 178.